26.12.03

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年１２月　１日

出願番号
Application Number: 特願２００３－４０２２０７

[ST. 10/C]: [ＪＰ２００３－４０２２０７]

出願人
Applicant(s): 松下電器産業株式会社

２００４年　２月　６日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫

出証番号　出証特２００４－３００７３２８

【書類名】　特許願
【整理番号】　2968150055
【提出日】　平成15年12月 1日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 9/45
【発明者】
　【住所又は居所】　愛知県名古屋市中区栄２丁目６番１号　白川ビル別館５階　株式会社松下電器情報システム名古屋研究所内
　【氏名】　河合　正樹
【発明者】
　【住所又は居所】　愛知県名古屋市中区栄２丁目６番１号　白川ビル別館５階　株式会社松下電器情報システム名古屋研究所内
　【氏名】　川本　琢二
【発明者】
　【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
　【氏名】　春名　修介
【発明者】
　【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
　【氏名】　藤原　寛
【特許出願人】
　【識別番号】　000005821
　【住所又は居所】　大阪府門真市大字門真１００６番地
　【氏名又は名称】　松下電器産業株式会社
【代理人】
　【識別番号】　100067828
　【弁理士】
　【氏名又は名称】　小谷　悦司
【選任した代理人】
　【識別番号】　100075409
　【弁理士】
　【氏名又は名称】　植木　久一
【選任した代理人】
　【識別番号】　100109438
　【弁理士】
　【氏名又は名称】　大月　伸介
【先の出願に基づく優先権主張】
　【出願番号】　特願2003-　275
　【出願日】　平成15年 1月 6日
【手数料の表示】
　【予納台帳番号】　012472
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　特許請求の範囲　1
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
　【包括委任状番号】　0214505

【書類名】特許請求の範囲
【請求項１】
原始プログラムを目的プログラムに変換するためのコンパイラプログラムであって、
前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成手段と、
前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成手段と、
前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成手段と、
前記移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成手段と、
前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成手段としてコンピュータを機能させることを特徴とするコンパイラプログラム。
【請求項２】
前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かを判別する判別手段としてコンピュータをさらに機能させ、
前記判別手段によって処理を短縮しないと判別された場合、
前記アドレス保存プログラム生成手段は、前記呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成し、
前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成し、
前記移行プログラム生成手段は、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムを生成し、
前記アドレス再設定プログラム生成手段は、前記移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成し、
前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成し、
前記判別手段によって処理を短縮すると判別された場合、
前記移行プログラム生成手段は、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムを生成し、
前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際

に、前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することを特徴とする請求項１記載のコンパイラプログラム。

【請求項３】

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムが呼び出された後、

前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムを生成し、

前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することを特徴とする請求項１記載のコンパイラプログラム。

【請求項４】

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かを判別する判別手段としてコンピュータをさらに機能させ、

前記判別手段によって処理を短縮しないと判別された場合、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムが呼び出された後、

前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムを生成し、

前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成し、

前記判別手段によって処理を短縮すると判別された場合、

前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することを特徴とする請求項３記載のコンパイラプログラム。

【請求項５】

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールの記述は、

前記第２のサブプログラム毎に記述される宣言であることを特徴とする請求項２又は４に記載のコンパイラプログラム。

【請求項６】

前記呼び出し元プログラムモジュールによって呼び出される第２のサブプログラムを含む他プログラムモジュールを含む前記原始プログラムは、

前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの複数の実行によって共通にアクセスする前記他プログラムモジュール用の実行共有デ

ータメモリ領域、及び前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの実行毎に固有にアクセスする前記他プログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのかを特定するための記述を含むことを特徴とする請求項１～５のいずれかに記載のコンパイラプログラム。

【請求項７】

前記他プログラムモジュール用の実行共有データメモリ領域、及び前記他プログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのかを特定するための記述は、

前記他プログラムモジュールに含まれる外部変数毎に記述される宣言であることを特徴とする請求項６に記載のコンパイラプログラム。

【請求項８】

前記アドレス保存プログラム生成手段は、前記呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを保存するとともに、前記呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成し、

前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用の実行共有データメモリ領域アドレスを設定するとともに、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用の実行固有データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成し、

前記移行プログラム生成手段は、前記第１のサブプログラムから前記第２のサブプログラムへ移行するための移行プログラムを生成し、

前記アドレス再設定プログラム生成手段は、前記移行された第２のサブプログラムから前記第１のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して再設定するとともに、前記保存された呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成し、

前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行固有データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするとともに、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行共有データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することを特徴とする請求項６に記載のコンパイラプログラム。

【請求項９】

前記原始プログラムに含まれる呼び出し元プログラムモジュールから前記他プログラムモジュールが呼び出された後、

前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して設定するとともに、前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムを生成し、

前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行固有データメモリ領域をアクセスする際に、前記設定された他プログラム用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするとともに、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行共有

データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することを特徴とする請求項６記載のコンパイラプログラム。

【請求項１０】

前記アドレス保存プログラム生成手段は、前記第１のサブプログラムから第２のサブプログラムへ移行するために必要なデータよりも先に、前記呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレス、及び実行固有データメモリ領域アドレスをスタックメモリに保存するためのアドレス保存プログラムを生成することを特徴とする請求項１に記載のコンパイラプログラム。

【請求項１１】

前記原始プログラムに含まれる前記呼び出し元プログラムモジュールと、

前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールとは、同じプログラムモジュールであることを特徴とする請求項１～１０のいずれかに記載のコンパイラプログラム。

【請求項１２】

前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、

前記呼び出し元プログラムモジュールから前記他プログラムモジュールへの呼び出し命令を受けて、前記他プログラムモジュールを識別する識別番号を取得するための識別番号取得プログラムを生成する識別番号取得プログラム生成手段と、

前記取得された識別番号をインデクスとして、前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとが対応付けられるとともに、前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとが対応付けられたテーブルから前記他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムを生成する先頭アドレス取得プログラム生成手段と、

前記取得された前記他プログラムモジュールのデータ領域の先頭アドレスと、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムを生成するための先頭アドレス切替プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項１記載のコンパイラプログラム。

【請求項１３】

前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、

前記呼び出し元プログラムモジュールを識別する識別番号と、前記他プログラムモジュールを識別する識別番号とをそれぞれに付与するための識別番号付与プログラムを生成する識別番号付与プログラム生成手段と、

前記付与された前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、前記付与された前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するためのテーブル作成プログラムを生成するテーブル作成プログラム生成手段と、

前記呼び出し元プログラムモジュールから前記他プログラムモジュールへの呼び出し命令を受けて、前記他プログラムモジュールの識別番号を取得するための識別番号取得プログラムを生成する識別番号取得プログラム生成手段と、

前記取得された識別番号をインデクスとして前記作成されたテーブルから前記他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムを生成する先頭アドレス取得プログラム生成手段と、

前記取得された前記他プログラムモジュールのデータ領域の先頭アドレスと、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アド

レス切替プログラムを生成するための先頭アドレス切替プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項１記載のコンパイラプログラム。

【請求項１４】

前記他プログラムモジュール外で定義された変数又は関数へのアドレスを保持するためのアドレス保持プログラムを生成するアドレス保持プログラム生成手段としてコンピュータをさらに機能させ、

前記アクセスプログラム生成手段は、前記他プログラムモジュール内で定義された変数へアクセスする場合、前記取得された先頭アドレスからの相対アドレスでアクセスし、前記他プログラムモジュール内で定義された関数へアクセスする場合、プログラムカウンタからの相対アドレスでアクセスし、前記他プログラムモジュール外で定義された変数又は関数へアクセスする場合、前記保持されたアドレスに前記取得された先頭アドレスからの相対アドレスでアクセスした後、前記保持されたアドレス経由で間接アクセスするためのアクセスプログラムを生成することを特徴とする請求項１記載のコンパイラプログラム。

【請求項１５】

少なくとも１つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムを生成する終了プログラム生成手段と、

アプリケーションが終了された後、メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、

メモリがコンパクションされた後、前記アプリケーションを最初から再開するための再開プログラムを生成する再開プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項１記載のコンパイラプログラム。

【請求項１６】

少なくとも１つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムを生成する終了プログラム生成手段と、

終了されるまでの前記アプリケーションのアドレス値に依存しない情報を保存するための保存プログラムを生成する保存プログラム生成手段と、

メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、

メモリがコンパクションされた後、前記保存された情報を読み出し、読み出された情報に基づいて前記アプリケーションが終了した時点における状態まで当該アプリケーションを実行するための実行プログラムを生成する実行プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項１記載のコンパイラプログラム。

【請求項１７】

メモリ内において、アドレス値を設定するか否かを示す識別子を設け、

コンパクション前のメモリの状態を記憶するためのメモリ状態記憶プログラムを生成するメモリ状態記憶プログラム生成手段と、

メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、

メモリがコンパクションされた後、前記識別子に基づいてアドレス値が設定されているエントリに対して、前記記憶されているコンパクション前のメモリの状態を参照して、アドレス値の再配置を行うための再配置プログラムを生成する再配置プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項１記載のコンパイラプログラム。

【請求項１８】

アドレス値を格納するアドレス値用メモリと、アドレス値以外を格納する非アドレス値用メモリとを設け、

コンパクション前のアドレス値用メモリ及び非アドレス値用メモリの状態を記憶するためのメモリ状態記憶プログラムを生成するメモリ状態記憶プログラム生成手段と、

前記アドレス値用メモリ及び前記非アドレス値用メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、

前記アドレス値用メモリ及び前記非アドレス値用メモリがコンパクションされた後、前記記憶されているコンパクション前のアドレス値用メモリの状態を参照して、前記アドレス値用メモリのみのアドレス値の再配置を行うための再配置プログラムを生成する再配置プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項１記載のコンパイラプログラム。

【請求項１９】

メモリ内において、モジュールを識別するための識別番号と、前記モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別子とを設け、

ポインタにアドレス値を代入する際に前記識別番号と前記識別子とを設定するための設定プログラムを生成する設定プログラム生成手段と、

前記メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、

前記メモリがコンパクションされた後、前記識別番号及び前記識別子に基づいてアドレス値を再計算するための再計算プログラムを生成する再計算プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項１記載のコンパイラプログラム。

【請求項２０】

メモリ内において、モジュールを識別するための識別番号と、前記モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別子とを設け、

ポインタにアドレス値を代入する際に前記識別番号と前記識別子とを設定するとともに、データ領域の先頭アドレス又はコード領域の先頭アドレスからのオフセット値を設定するためのオフセット値設定プログラムを生成するオフセット値設定プログラム生成手段と、

前記ポインタを使用する際に、前記オフセット値にデータ領域の先頭アドレス又はコード領域の先頭アドレスを加算することによって実アドレスに変換するための実アドレス変換プログラムを生成する実アドレス変換プログラム生成手段としてコンピュータをさらに機能させることを特徴とする請求項１記載のコンパイラプログラム。

【請求項２１】

モジュールのコード領域又はデータ領域毎にコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、

コンパクションによって移動する前記コード領域又は前記データ領域を指すハンドルがあるか否かを検索するための検索プログラムを生成する検索プログラム生成手段と、

前記コード領域又は前記データ領域を指すハンドルが発見された場合、当該ハンドルを再配置するためのハンドル再配置プログラムを生成するハンドル再配置プログラム生成手段としてコンピュータをさらに機能させ、

前記コンパクションプログラム生成手段は、全てのハンドルについて再配置を行った後、当該コード領域又はデータ領域をコンパクションするためのコンパクションプログラムを生成することを特徴とする請求項１記載のコンパイラプログラム。

【請求項２２】

原始プログラムを目的プログラムに変換するためのコンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成手段と、

前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成手段と、

前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムを生

成する移行プログラム生成手段と、
前記移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成手段と、
前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成手段としてコンピュータを機能させることを特徴とするコンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体。
【請求項２３】
原始プログラムを目的プログラムに変換するためのコンパイル方法であって、
コンピュータが、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成ステップと、
コンピュータが、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成ステップと、
コンピュータが、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成ステップと、
コンピュータが、前記移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成ステップと、
コンピュータが、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成ステップとを含むことを特徴とするコンパイル方法。
【請求項２４】
原始プログラムを目的プログラムに変換するコンパイル装置であって、
前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成手段と、
前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成手段と、
前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成手段と、
前記移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成手段と、
前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成手段とを備えることを

特徴とするコンパイル装置。

【請求項２５】

原始プログラムが変換されることによって生成される目的プログラムであって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存手段と、

前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定手段と、

前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行する移行手段と、

前記移行手段によって移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記アドレス保存手段によって保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定手段と、

前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定手段によって設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセス手段としてコンピュータを機能させることを特徴とする目的プログラム。

【請求項２６】

原始プログラムが変換されることによって生成される目的プログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存手段と、

前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定手段と、

前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行する移行手段と、

前記移行手段によって移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記アドレス保存手段によって保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定手段と、

前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定手段によって設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセス手段としてコンピュータを機能させることを特徴とする目的プログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項２７】

原始プログラムが変換されることによって生成される目的プログラムを実行する目的プログラム実行方法であって、

コンピュータが、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存ステップと、

コンピュータが、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定ステップと、

コンピュータが、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行する移行ステップと、

コンピュータが、前記移行ステップにおいて移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記アドレス保存ステップにおいて保存された呼び出し

元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定ステップと、

コンピュータが、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定ステップにおいて設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセスステップとを含むことを特徴とする目的プログラム実行方法。

【請求項２８】

原始プログラムが変換されることによって生成される目的プログラムを実行する目的プログラム実行装置であって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存手段と、

前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定手段と、

前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行する移行手段と、

前記移行手段によって移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記アドレス保存手段によって保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定手段と、

前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定手段によって設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセス手段とを備えることを特徴とする目的プログラム実行装置。

【請求項２９】

前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、

前記呼び出し元プログラムモジュールを識別する識別番号と、前記他プログラムモジュールを識別する識別番号とをそれぞれに付与する識別番号付与手段と、

前記識別番号付与手段によって付与された前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、前記識別番号付与手段によって付与された前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するテーブル作成手段と、

前記呼び出し元プログラムモジュールから前記他プログラムモジュールへの呼び出し命令を受けて、前記他プログラムモジュールの識別番号を取得する識別番号取得手段と、

前記識別番号取得手段によって取得された識別番号をインデクスとして、前記テーブル作成手段によって作成されたテーブルから前記他プログラムモジュールのデータ領域の先頭アドレスを取得する先頭アドレス取得手段と、

前記先頭アドレス取得手段によって取得された前記他プログラムモジュールのデータ領域の先頭アドレスと、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替える先頭アドレス切替手段とをさらに備えることを特徴とする請求項２８記載の目的プログラム実行装置。

【請求項３０】

前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、

前記呼び出し元プログラムモジュールを識別する識別番号と、前記他プログラムモジュールを識別する識別番号とをそれぞれに付与する識別番号付与手段と、

前記識別番号付与手段によって付与された前記呼び出し元プログラムモジュールの識別

番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、前記識別番号付与手段によって付与された前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するテーブル作成手段とをさらに備えることを特徴とする請求項２８記載の目的プログラム実行装置。

【書類名】明細書
【発明の名称】コンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置
【技術分野】
【０００１】
本発明は、あるプログラム言語で記述された原始プログラム（以後、「ソースプログラム」または「ソースプログラムコード」または「ソースコード」とも言う）を、あるコンピュータによって実行するための目的プログラム（「以後、オブジェクトプログラム」または「オブジェクトコードプログラム」または「オブジェクトコード」とも言う）に変換（以後場合によっては「翻訳」と言うこともあるが同一内容を示す）するためのコンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置に関するものである。
【０００２】
また本特許出願の明細書で言う、目的プログラムまたはオブジェクトプログラム等には、コンピュータが直接実行可能なプログラムコードに加え、コンピュータが直接実行可能なプログラムコードに変換する１段階である中間プログラムコードをも含むものであり、またそれに加え特に、コンピュータが直接実行可能なプログラムコードでは確定するべき実際にアクセスされるメモリアドレスが確定していない段階の中間コードプログラムをも含むものであり、更にまたそれに加え特に、コンピュータプラグラムが直接実行可能なプログラムコード全体を一体化する以前の段階の中間コードプログラムをも含むものである。
【０００３】
また例えば、いわゆる機器組み込みプログラムのようにオブジェクトプログラムを格納することのできるメモリ領域が限られており、しかも近年の携帯情報機器のようにその限られたメモリ領域に高度且つ高機能な情報処理のためのオブジェクトプログラムを格納しなければならないような目的に適う、オブジェクトプログラムを生成するためのコンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置に関するものである。
【０００４】
更にまた、このような目的に適うオブジェクトプログラムとして再入可能（以後、「リエントラント：ｒｅｅｎｔｒａｎｔ」と言うこともある）なオブジェクトプログラムを自動生成するためのコンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置に関するものである。
【背景技術】
【０００５】
従来、時分割または優先度プリエンプティブタスクまたは、プロセススケジューリング機能、及びそれに類する機能を有するオペレーティングシステムや、関数型プログラミング言語を用いることを条件として制作されるプログラムであって、複数のタスクまたはプロセスからある一定の時間間隔の中で、同時に呼び出される可能性がある関数プログラムは再入可能であることが要求された。
【０００６】
そしてこのような再入可能関数プログラムは、人間のプログラマによってコーディングされ、再入可能性が満足されているか否かを自動的に判定するためのプログラムが作成されていた（例えば、特許文献１参照）。
【０００７】
この再入可能性を満足する条件は、判定を行う対象である関数を記述しているソースプログラムの全体を構文解析してその全フローを抽出し、その全フローでアクセスされるすべてのリソースに対して、スコープを決定し、そのリソースのスコープの種類に応じて完全な排他的制御が行われているか否かに依存するものであった。
【特許文献１】特開２００１－１３４４３１号公報

【発明の開示】
【発明が解決しようとする課題】
【０００８】
しかしながら、ソースプログラムの全体を構文解析してその全フローを抽出し、その全フローでアクセスされるすべてのリソースに対して、スコープを決定し、そのリソースのスコープの種類に応じて完全な排他的制御が行われているか否かを完全に行うことは時間的な制約やコンピュータの処理能力の限界から、現実には非常に困難で、ある一定レベルの判定に留まらざるを得なかった。
【０００９】
そして、従来の再入可能なプログラムの作成は、その本質的部分が人間のプログラマの作業によって記述されるため、多くの時間を要しながら不完全な排他的制御しか実現することができず、その結果、プログラムを実行させた時に多くの誤動作を引き起こす原因となっていた。
【００１０】
また、このような再入可能なプログラムはマルチタスクや時分割環境下で使用されることが多いため、その原因発見は極めて困難であった。更にまた、このような再入可能なプログラムの更新や改訂を行うことは、更に一層困難なものであった。
【００１１】
本発明は、上記の課題を解決するためになされたもので、自動的に再入可能な目的プログラムを生成することができるコンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置を提供することを目的とするものである。
【課題を解決するための手段】
【００１２】
本発明に係るコンパイラプログラムは、原始プログラムを目的プログラムに変換するためのコンパイラプログラムであって、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成手段と、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成手段と、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成手段と、前記移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成手段と、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成手段としてコンピュータを機能させる。
【００１３】
この構成によれば、アドレス保存プログラム生成手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムが生成される。そして、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムが生成される。さらに、移行プログラム生成手段によって、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プ

ログラムが生成される。アドレス再設定プログラム生成手段によって、移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムが生成される。アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【００１４】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【００１５】

また、上記のコンパイラプログラムにおいて、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かを判別する判別手段としてコンピュータをさらに機能させ、前記判別手段によって処理を短縮しないと判別された場合、前記アドレス保存プログラム生成手段は、前記呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成し、前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成し、前記移行プログラム生成手段は、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムを生成し、前記アドレス再設定プログラム生成手段は、前記移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成し、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成し、前記判別手段によって処理を短縮すると判別された場合、前記移行プログラム生成手段は、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムを生成し、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することが好ましい。

【００１６】

この構成によれば、判別手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かが判別される。判別手段によって処理を短縮しないと判別された場合、アドレス保存プログラム生成手段によって、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムが生成される。そし

て、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムが生成される。さらに、移行プログラム生成手段によって、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムが生成される。また、アドレス再設定プログラム生成手段によって、移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムが生成される。さらにまた、アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【００１７】

一方、判別手段によって処理を短縮すると判別された場合、移行プログラム生成手段によって、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムが生成される。そして、アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【００１８】

したがって、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かが判別されるので、必ずしも呼び出し元プログラムモジュールと呼び出される他プログラムモジュールとが異なるデータ領域を使用することが必要でないケースでは、アドレスの保存や設定や再設定の処理を省略し、目的プログラムの実行を高速化することができ、不要な処理を出力しないことで目的プログラムのサイズを小さくすることができる。

【００１９】

また、上記のコンパイラプログラムにおいて、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムが呼び出された後、前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムを生成し、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することが好ましい。

【００２０】

この構成によれば、原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムが呼び出された後、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから他プログラムモジュール用データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムが生成される。そして、アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのア

クセスプログラムが生成される。

【００２１】

したがって、呼び出し元プログラムモジュールから他プログラムモジュールへの再入が発生したとしても、各プログラムモジュール用データメモリ領域アドレステーブルを介してアクセスするので、例えば外部変数へのアクセスで衝突が発生することは起こらず、プログラムモジュールを再入可能に構成することができる。

【００２２】

また、上記のコンパイラプログラムにおいて、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かを判別する判別手段としてコンピュータをさらに機能させ、前記判別手段によって処理を短縮しないと判別された場合、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムが呼び出された後、前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムを生成し、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成し、前記判別手段によって処理を短縮すると判別された場合、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することが好ましい。

【００２３】

この構成によれば、判別手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かが判別される。判別手段によって処理を短縮しないと判別された場合、原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムが呼び出された後、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから他プログラムモジュール用データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムが生成される。そして、アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【００２４】

一方、判別手段によって処理を短縮すると判別された場合、アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【００２５】

したがって、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールの記述に基づいて、処理を短縮するか否かが判別されるので、必ずしも呼び出し元プログラムモジュールと呼び出

される他プログラムモジュールとが異なるデータ領域を使用することが必要でないケースでは、各プログラムモジュール用データメモリ領域アドレステーブルを介したアドレスの設定の処理を省略し、目的プログラムの実行を高速化することができ、不要な処理を出力しないことで目的プログラムのサイズを小さくすることができる。

【００２６】

また、上記のコンパイラプログラムにおいて、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールの記述は、前記第２のサブプログラム毎に記述される宣言であることが好ましい。

【００２７】

この構成によれば、原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールの記述が、第２のサブプログラム毎に記述される宣言であるので、この宣言の有無により、第２のサブプログラムの処理内容に応じてアドレスの設定の処理を省略することができ、目的プログラムの実行を高速化することができる。

【００２８】

また、上記のコンパイラプログラムにおいて、前記呼び出し元プログラムモジュールによって呼び出される第２のサブプログラムを含む他プログラムモジュールを含む前記原始プログラムは、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの複数の実行によって共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域、及び前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの実行毎に固有にアクセスする前記他プログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのかを特定するための記述を含むことが好ましい。

【００２９】

この構成によれば、呼び出し元プログラムモジュールによって呼び出される第２のサブプログラムを含む他プログラムモジュールを含む原始プログラムに含まれる記述によって、他プログラムモジュールに含まれる外部変数が呼び出し元プログラムモジュールの複数の実行によって共通にアクセスする他プログラムモジュール用の実行共有データメモリ領域、及び他プログラムモジュールに含まれる外部変数が呼び出し元プログラムモジュールの実行毎に固有にアクセスする他プログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのかが特定される。

【００３０】

したがって、他プログラムモジュールが複数の呼び出し元プログラムモジュールから呼び出されたとしても、各呼び出し元プログラムモジュール毎に設けられた実行固有データメモリ領域にアクセスするため、例えば外部変数へのアクセスで衝突が発生することは起こらず、プログラムモジュールを再入可能に構成することができる。また、他プログラムモジュールが複数の呼び出し元プログラムモジュールから呼び出されたとしても、各呼び出し元プログラムモジュール毎で共有する実行共有データメモリ領域にアクセスするため、例えば外部変数の幾つかを共通に利用することができる。

【００３１】

また、上記のコンパイラプログラムにおいて、前記他プログラムモジュール用の実行共有データメモリ領域、及び前記他プログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのかを特定するための記述は、前記他プログラムモジュールに含まれる外部変数毎に記述される宣言であることが好ましい。

【００３２】

この構成によれば、他プログラムモジュール用の実行共有データメモリ領域、及び他プログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのかを特定するための記述が、他プログラムモジュールに含まれる外部変数毎に記述される宣言であるので、この宣言の有無を判断することによって容易に実行共有データメモリ領域と実行

固有データメモリ領域とを使い分けることができる。

【００３３】

また、上記のコンパイラプログラムにおいて、前記アドレス保存プログラム生成手段は、前記呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを保存するとともに、前記呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成し、前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用の実行共有データメモリ領域アドレスを設定するとともに、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用の実行固有データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成し、前記移行プログラム生成手段は、前記第１のサブプログラムから前記第２のサブプログラムへ移行するための移行プログラムを生成し、前記アドレス再設定プログラム生成手段は、前記移行された第２のサブプログラムから前記第１のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して再設定するとともに、前記保存された呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成し、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行固有データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするとともに、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行共有データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することが好ましい。

【００３４】

この構成によれば、アドレス保存プログラム生成手段によって、呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを保存するとともに、呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを保存するためのアドレス保存プログラムが生成される。そして、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用の実行共有データメモリ領域アドレスを設定するとともに、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用の実行固有データメモリ領域アドレスを設定するためのアドレス設定プログラムが生成される。移行プログラム生成手段によって、第１のサブプログラムから第２のサブプログラムへ移行するための移行プログラムが生成される。アドレス再設定プログラム生成手段によって、移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して再設定するとともに、保存された呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムが生成される。アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用の実行固有データメモリ領域をアクセスする際に、設定された他プログラムモジュール用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするとともに、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用の実行共有データメモリ領域をアクセスする際に、設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【００３５】

したがって、実行固有データメモリ領域と実行共有データメモリ領域とを使い分けることができ、他プログラムモジュールが複数の呼び出し元プログラムモジュールによって呼び出されたとしても、例えば外部変数へのアクセスで衝突が発生することは起こらず、プ

ログラムモジュールを再入可能に構成することができ、さらに、複数のプログラムモジュールは外部変数を共有することができる。

【００３６】

また、上記のコンパイラプログラムにおいて、前記原始プログラムに含まれる呼び出し元プログラムモジュールから前記他プログラムモジュールが呼び出された後、前記アドレス設定プログラム生成手段は、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して設定するとともに、前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムを生成し、前記アクセスプログラム生成手段は、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行固有データメモリ領域をアクセスする際に、前記設定された他プログラム用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするとともに、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用の実行共有データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成することが好ましい。

【００３７】

この構成によれば、原始プログラムに含まれる呼び出し元プログラムモジュールから他プログラムモジュールが呼び出された後、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから他プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して設定するとともに、呼び出し元プログラムモジュールの複数の実行が共通にアクセスする他プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して設定するためのアドレス設定プログラムが生成される。アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用の実行固有データメモリ領域をアクセスする際に、設定された他プログラム用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするとともに、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用の実行共有データメモリ領域をアクセスする際に、設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【００３８】

したがって、実行固有データメモリ領域と実行共有データメモリ領域とを使い分けることができ、他プログラムモジュールが複数の呼び出し元プログラムモジュールによって呼び出されたとしても、例えば外部変数へのアクセスで衝突が発生することは起こらず、プログラムモジュールを再入可能に構成することができ、さらに、複数のプログラムモジュールは外部変数を共有することができる。

【００３９】

また、上記のコンパイラプログラムにおいて、前記アドレス保存プログラム生成手段は、前記第１のサブプログラムから第２のサブプログラムへ移行するために必要なデータよりも先に、前記呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレス、及び実行固有データメモリ領域アドレスをスタックメモリに保存するためのアドレス保存プログラムを生成することが好ましい。

【００４０】

この構成によれば、アドレス保存プログラム生成手段によって、第１のサブプログラムから第２のサブプログラムへ移行するために必要なデータよりも先に、呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレス、及び実行固有データメモリ領域アドレスをスタックメモリに保存するためのアドレス保存プログラムが生成される。

【００４１】
したがって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール側が、スタックメモリに保存された実行共有データメモリ領域アドレス及び実行固有データメモリ領域アドレスの有無に関与する必要をなくすことができる。
【００４２】
また、上記のコンパイラプログラムにおいて、前記原始プログラムに含まれる前記呼び出し元プログラムモジュールと、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールとは、同じプログラムモジュールであることが好ましい。
【００４３】
この構成によれば、原始プログラムに含まれる呼び出し元プログラムモジュールと、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから呼び出される第２のサブプログラムを含む他プログラムモジュールとが、同じプログラムモジュールである。
【００４４】
したがって、同じモジュール内の呼び出しを短縮することによって、モジュール内の処理を低速化させずに済むことができる。
【００４５】
また、上記のコンパイラプログラムにおいて、前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、前記呼び出し元プログラムモジュールから前記他プログラムモジュールへの呼び出し命令を受けて、前記他プログラムモジュールを識別する識別番号を取得するための識別番号取得プログラムを生成する識別番号取得プログラム生成手段と、前記取得された識別番号をインデクスとして、前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとが対応付けられるとともに、前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとが対応付けられたテーブルから前記他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムを生成する先頭アドレス取得プログラム生成手段と、前記取得された前記他プログラムモジュールのデータ領域の先頭アドレスと、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムを生成するための先頭アドレス切替プログラム生成手段としてコンピュータをさらに機能させることが好ましい。
【００４６】
この構成によれば、呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、他プログラムモジュールはコード領域とデータ領域とを含み、識別番号取得プログラム生成手段によって、呼び出し元プログラムモジュールから他プログラムモジュールへの呼び出し命令を受けて、他プログラムモジュールを識別する識別番号を取得するための識別番号取得プログラムが生成される。先頭アドレス取得プログラム生成手段によって、取得された識別番号をインデクスとして、呼び出し元プログラムモジュールの識別番号と、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとが対応付けられるとともに、他プログラムモジュールの識別番号と、他プログラムモジュールのデータ領域の先頭アドレスとが対応付けられたテーブルから他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムが生成される。そして、先頭アドレス切替プログラム生成手段によって、取得された他プログラムモジュールのデータ領域の先頭アドレスと、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムが生成される。
【００４７】
したがって、アドレス変換するためのテーブルを、ページ単位ではなくモジュール単位で行うことによって、テーブルのデータ量を削減することができ、資源制約の厳しい小型の機器に適合させることができる。
【００４８】

また、上記のコンパイラプログラムにおいて、前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、前記呼び出し元プログラムモジュールを識別する識別番号と、前記他プログラムモジュールを識別する識別番号とをそれぞれに付与するための識別番号付与プログラムを生成する識別番号付与プログラム生成手段と、前記付与された前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、前記付与された前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するためのテーブル作成プログラムを生成するテーブル作成プログラム生成手段と、前記呼び出し元プログラムモジュールから前記他プログラムモジュールへの呼び出し命令を受けて、前記他プログラムモジュールの識別番号を取得するための識別番号取得プログラムを生成する識別番号取得プログラム生成手段と、前記取得された識別番号をインデクスとして前記作成されたテーブルから前記他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムを生成する先頭アドレス取得プログラム生成手段と、前記取得された前記他プログラムモジュールのデータ領域の先頭アドレスと、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムを生成するための先頭アドレス切替プログラム生成手段としてコンピュータをさらに機能させることが好ましい。

【００４９】

この構成によれば、呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、他プログラムモジュールはコード領域とデータ領域とを含み、識別番号付与プログラム生成手段によって、呼び出し元プログラムモジュールを識別する識別番号と、他プログラムモジュールを識別する識別番号とをそれぞれに付与するための識別番号付与プログラムが生成される。テーブル作成プログラム生成手段によって、付与された呼び出し元プログラムモジュールの識別番号と、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、付与された他プログラムモジュールの識別番号と、他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するためのテーブル作成プログラムが生成される。識別番号取得プログラム生成手段によって、呼び出し元プログラムモジュールから他プログラムモジュールへの呼び出し命令を受けて、他プログラムモジュールの識別番号を取得するための識別番号取得プログラムが生成される。先頭アドレス取得プログラム生成手段によって、取得された識別番号をインデクスとして、作成されたテーブルから他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムが生成される。先頭アドレス切替プログラム生成手段によって、取得された他プログラムモジュールのデータ領域の先頭アドレスと、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムが生成される。

【００５０】

したがって、アドレス変換するためのテーブルを、ページ単位ではなくモジュール単位で行うことによって、テーブルのデータ量を削減することができ、資源制約の厳しい小型の機器に適合させることができる。

【００５１】

また、上記のコンパイラプログラムにおいて、前記他プログラムモジュール外で定義された変数又は関数へのアドレスを保持するためのアドレス保持プログラムを生成するアドレス保持プログラム生成手段としてコンピュータをさらに機能させ、前記アクセスプログラム生成手段は、前記他プログラムモジュール内で定義された変数へアクセスする場合、前記取得された先頭アドレスからの相対アドレスでアクセスし、前記他プログラムモジュール内で定義された関数へアクセスする場合、プログラムカウンタからの相対アドレスでアクセスし、前記他プログラムモジュール外で定義された変数又は関数へアクセスする場合、前記保持されたアドレスに前記取得された先頭アドレスからの相対アドレスでアクセスした後、前記保持されたアドレス経由で間接アクセスするためのアクセスプログラムを

生成することが好ましい。

【００５２】

この構成によれば、アドレス保持プログラム生成手段によって、他プログラムモジュール外で定義された変数又は関数へのアドレスを保持するためのアドレス保持プログラムが生成される。そして、アクセスプログラム生成手段によって、他プログラムモジュール内で定義された変数又は関数へアクセスする場合、取得された先頭アドレスからの相対アドレスでアクセスされ、他プログラムモジュール内で定義された関数へアクセスする場合、プログラムカウンタからの相対アドレスでアクセスされ、他プログラムモジュール外で定義された変数又は関数へアクセスする場合、保持されたアドレスに取得された先頭アドレスからの相対アドレスでアクセスした後、保持されたアドレス経由で間接アクセスするためのアクセスプログラムが生成される。したがって、他プログラムモジュール外で定義された変数又は関数へアクセスする場合に、保持されたアドレス経由で間接的にアクセスすることができる。

【００５３】

また、上記のコンパイラプログラムにおいて、少なくとも１つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムを生成する終了プログラム生成手段と、アプリケーションが終了された後、メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、メモリがコンパクションされた後、前記アプリケーションを最初から再開するための再開プログラムを生成する再開プログラム生成手段としてコンピュータをさらに機能させることが好ましい。

【００５４】

この構成によれば、終了プログラム生成手段によって、少なくとも１つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムが生成される。そして、アプリケーションが終了された後、コンパクションプログラム生成手段によって、メモリをコンパクションするためのコンパクションプログラムが生成され、メモリがコンパクションされた後、再開プログラム生成手段によって、アプリケーションを最初から再開するための再開プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【００５５】

また、上記のコンパイラプログラムにおいて、少なくとも１つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムを生成する終了プログラム生成手段と、終了されるまでの前記アプリケーションのアドレス値に依存しない情報を保存するための保存プログラムを生成する保存プログラム生成手段と、メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、メモリがコンパクションされた後、前記保存された情報を読み出し、読み出された情報に基づいて前記アプリケーションが終了した時点における状態まで当該アプリケーションを実行するための実行プログラムを生成する実行プログラム生成手段としてコンピュータをさらに機能させることが好ましい。

【００５６】

この構成によれば、終了プログラム生成手段によって、少なくとも１つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムが生成される。そして、保存プログラム生成手段によって、終了されるまでのアプリケーションのアドレス値に依存しない情報を保存するための保存プログラムが生成される。コンパクションプログラム生成手段によって、メモリをコンパクションするためのコンパクションプログラムが生成される。メモリがコンパクションされた後、実行プログラム生成手段によって、保存された情報を読み出し、読み出された情報に基づいてアプリケーションが終了した時点における状態まで当該アプリケーションを実行するための実行プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【００５７】
また、上記のコンパイラプログラムにおいて、メモリ内において、アドレス値を設定するか否かを示す識別子を設け、コンパクション前のメモリの状態を記憶するためのメモリ状態記憶プログラムを生成するメモリ状態記憶プログラム生成手段と、メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、メモリがコンパクションされた後、前記識別子に基づいてアドレス値が設定されているエントリに対して、前記記憶されているコンパクション前のメモリの状態を参照して、アドレス値の再配置を行うための再配置プログラムを生成する再配置プログラム生成手段としてコンピュータをさらに機能させることが好ましい。
【００５８】
この構成によれば、メモリ内において、アドレス値を設定するか否かを示す識別子を設け、メモリ状態記憶プログラム生成手段によって、コンパクション前のメモリの状態を記憶するためのメモリ状態記憶プログラムが生成される。そして、コンパクションプログラム生成手段によって、メモリをコンパクションするためのコンパクションプログラムが生成される。さらに、メモリがコンパクションされた後、再配置プログラム生成手段によって、識別子に基づいてアドレス値が設定されているエントリに対して、記憶されているコンパクション前のメモリの状態を参照して、アドレス値の再配置を行うための再配置プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。
【００５９】
また、上記のコンパイラプログラムにおいて、アドレス値を格納するアドレス値用メモリと、アドレス値以外を格納する非アドレス値用メモリとを設け、コンパクション前のアドレス値用メモリ及び非アドレス値用メモリの状態を記憶するためのメモリ状態記憶プログラムを生成するメモリ状態記憶プログラム生成手段と、前記アドレス値用メモリ及び前記非アドレス値用メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、前記アドレス値用メモリ及び前記非アドレス値用メモリがコンパクションされた後、前記記憶されているコンパクション前のアドレス値用メモリの状態を参照して、前記アドレス値用メモリのみのアドレス値の再配置を行うための再配置プログラムを生成する再配置プログラム生成手段としてコンピュータをさらに機能させることが好ましい。
【００６０】
この構成によれば、アドレス値を格納するアドレス値用メモリと、アドレス値以外を格納する非アドレス値用メモリとを設け、アドレス値用メモリ状態記憶プログラム生成手段によって、コンパクション前のアドレス値用メモリ及び非アドレス値用メモリの状態を記憶するためのアドレス値用メモリ状態記憶プログラムが生成される。コンパクションプログラム生成手段によって、アドレス値用メモリ及び非アドレス値用メモリをコンパクションするためのコンパクションプログラムが生成される。アドレス値用メモリ及び非アドレス値用メモリがコンパクションされた後、再配置プログラム生成手段によって、記憶されているコンパクション前のアドレス値用メモリの状態を参照して、アドレス値用メモリのみのアドレス値の再配置を行うための再配置プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。
【００６１】
また、上記のコンパイラプログラムにおいて、メモリ内において、モジュールを識別するための識別番号と、前記モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別子とを設け、ポインタにアドレス値を代入する際に前記識別番号と前記識別子とを設定するための設定プログラムを生成する設定プログラム生成手段と、前記メモリをコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、前記メモリがコンパクションされた後、前記識別番号及び前記識別子に基づいてアドレス値を再計算するための再計算プログラム

を生成する再計算プログラム生成手段としてコンピュータをさらに機能させることが好ましい。

【００６２】

この構成によれば、メモリ内において、モジュールを識別するための識別番号と、モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別子とを設け、設定プログラム生成手段によって、ポインタにアドレス値を代入する際に識別番号と識別子とを設定するための設定プログラムが生成される。コンパクションプログラム生成手段によって、メモリをコンパクションするためのコンパクションプログラムが生成される。メモリがコンパクションされた後、再計算プログラム生成手段によって、識別番号及び識別子に基づいてアドレス値を再計算するための再計算プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【００６３】

また、上記のコンパイラプログラムにおいて、メモリ内において、モジュールを識別するための識別番号と、前記モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別子とを設け、ポインタにアドレス値を代入する際に前記識別番号と前記識別子とを設定するとともに、データ領域の先頭アドレス又はコード領域の先頭アドレスからのオフセット値を設定するためのオフセット値設定プログラムを生成するオフセット値設定プログラム生成手段と、前記ポインタを使用する際に、前記オフセット値にデータ領域の先頭アドレス又はコード領域の先頭アドレスを加算することによって実アドレスに変換するための実アドレス変換プログラムを生成する実アドレス変換プログラム生成手段としてコンピュータをさらに機能させることが好ましい。

【００６４】

この構成によれば、メモリ内において、モジュールを識別するための識別番号と、モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別子とを設け、オフセット値設定プログラム生成手段によって、ポインタにアドレス値を代入する際に識別番号と識別子とを設定するとともに、データ領域の先頭アドレス又はコード領域の先頭アドレスからのオフセット値を設定するためのオフセット値設定プログラムが生成される。ポインタを使用する際に、実アドレス変換プログラム生成手段によって、オフセット値にデータ領域の先頭アドレス又はコード領域の先頭アドレスを加算することによって実アドレスに変換するための実アドレス変換プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【００６５】

また、上記のコンパイラプログラムにおいて、モジュールのコード領域又はデータ領域毎にコンパクションするためのコンパクションプログラムを生成するコンパクションプログラム生成手段と、コンパクションによって移動する前記コード領域又は前記データ領域を指すハンドルがあるか否かを検索するための検索プログラムを生成する検索プログラム生成手段と、前記コード領域又は前記データ領域を指すハンドルが発見された場合、当該ハンドルを再配置するためのハンドル再配置プログラムを生成するハンドル再配置プログラム生成手段としてコンピュータをさらに機能させ、前記コンパクションプログラム生成手段は、全てのハンドルについて再配置を行った後、当該コード領域又はデータ領域をコンパクションするためのコンパクションプログラムを生成することが好ましい。

【００６６】

この構成によれば、検索プログラム生成手段によって、コンパクションによって移動するコード領域又はデータ領域を指すハンドルがあるか否かを検索するための検索プログラムが生成される。コード領域又はデータ領域を指すハンドルが発見された場合、ハンドル再配置プログラム生成手段によって、当該ハンドルを再配置するためのハンドル再配置プログラムが生成される。全てのハンドルについて再配置を行った後、コンパクションプログラム生成手段によって、当該コード領域又はデータ領域をコンパクションするためのコ

ンパクションプログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【００６７】

また、本発明に係るコンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体は、原始プログラムを目的プログラムに変換するためのコンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体であって、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成手段と、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成手段と、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成手段と、前記移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成手段と、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成手段としてコンピュータを機能させるコンパイラプログラムを記録している。

【００６８】

この構成によれば、アドレス保存プログラム生成手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムが生成される。そして、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムが生成される。さらに、移行プログラム生成手段によって、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムが生成される。アドレス再設定プログラム生成手段によって、移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムが生成される。アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【００６９】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【００７０】

また、本発明に係るコンパイル方法は、原始プログラムを目的プログラムに変換するためのコンパイル方法であって、コンピュータが、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域

アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成ステップと、コンピュータが、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成ステップと、コンピュータが、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成ステップと、コンピュータが、前記移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成ステップと、コンピュータが、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成ステップとを含む。

【００７１】

この構成によれば、アドレス保存プログラム生成ステップにおいて、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムが生成される。そして、アドレス設定プログラム生成ステップにおいて、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムが生成される。さらに、移行プログラム生成ステップにおいて、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムが生成される。アドレス再設定プログラム生成ステップにおいて、移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムが生成される。アクセスプログラム生成ステップにおいて、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【００７２】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【００７３】

また、本発明に係るコンパイル装置は、原始プログラムを目的プログラムに変換するコンパイル装置であって、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成するアドレス保存プログラム生成手段と、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成するアドレス設定プログラム生成手段と、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムを生成する移行プログラム生成手段と、前記移

行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成するアドレス再設定プログラム生成手段と、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成するアクセスプログラム生成手段とを備える。

【００７４】

この構成によれば、アドレス保存プログラム生成手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムが生成される。そして、アドレス設定プログラム生成手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムが生成される。さらに、移行プログラム生成手段によって、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムが生成される。アドレス再設定プログラム生成手段によって、移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムが生成される。アクセスプログラム生成手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムが生成される。

【００７５】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【００７６】

また、本発明に係る目的プログラムは、原始プログラムが変換されることによって生成される目的プログラムであって、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存手段と、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定手段と、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行する移行手段と、前記移行手段によって移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記アドレス保存手段によって保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定手段と、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定手段によって設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセス手段としてコンピュータを機能させる。

【００７７】

この構成によれば、アドレス保存手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域ア

ドレスが保存される。そして、アドレス設定手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスが設定される。さらに、移行手段によって、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムへ移行される。アドレス再設定手段によって、移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスが読み出されて再設定される。アクセス手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域がアクセスされる。

【００７８】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【００７９】

また、本発明に係る目的プログラムを記録したコンピュータ読み取り可能な記録媒体は、原始プログラムが変換されることによって生成される目的プログラムを記録したコンピュータ読み取り可能な記録媒体であって、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存手段と、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定手段と、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行する移行手段と、前記移行手段によって移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記アドレス保存手段によって保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定手段と、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定手段によって設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセス手段としてコンピュータを機能させることを特徴とする目的プログラムを記録している。

【００８０】

この構成によれば、アドレス保存手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存される。そして、アドレス設定手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスが設定される。さらに、移行手段によって、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムへ移行される。アドレス再設定手段によって、移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスが読み出されて再設定される。アクセス手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域がアクセスされる。

【００８１】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる

再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【００８２】

また、本発明に係る目的プログラム実行方法は、原始プログラムが変換されることによって生成される目的プログラムを実行する目的プログラム実行方法であって、コンピュータが、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存ステップと、コンピュータが、前記呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定ステップと、コンピュータが、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行する移行ステップと、コンピュータが、前記移行ステップにおいて移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記アドレス保存ステップにおいて保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定ステップと、コンピュータが、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定ステップにおいて設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセスステップとを含む。

【００８３】

この構成によれば、アドレス保存ステップにおいて、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存される。そして、アドレス設定ステップにおいて、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスが設定される。さらに、移行ステップにおいて、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムへ移行される。アドレス再設定ステップにおいて、移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスが読み出されて再設定される。アクセスステップにおいて、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域がアクセスされる。

【００８４】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【００８５】

また、本発明に係る目的プログラム実行装置は、原始プログラムが変換されることによって生成される目的プログラムを実行する目的プログラム実行装置であって、前記原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するアドレス保存手段と、前記呼び出し元プ

ログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するアドレス設定手段と、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムから前記他プログラムモジュールに含まれる第２のサブプログラムへ移行する移行手段と、前記移行手段によって移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、前記アドレス保存手段によって保存された呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するアドレス再設定手段と、前記他プログラムモジュールが前記他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスする際に、前記アドレス設定手段によって設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするアクセス手段とを備える。

【００８６】

この構成によれば、アドレス保存手段によって、原始プログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存される。そして、アドレス設定手段によって、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスが設定される。さらに、移行手段によって、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムへ移行される。アドレス再設定手段によって、移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメモリ領域アドレスが読み出されて再設定される。アクセス手段によって、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域がアクセスされる。

【００８７】

したがって、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスが保存されるので、呼び出し元プログラムモジュール用データメモリ領域アドレスを退避させることができ、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュール用データメモリ領域アドレスが設定されるので、再入可能な目的プログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【００８８】

また、上記の目的プログラム実行装置において、前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、前記呼び出し元プログラムモジュールを識別する識別番号と、前記他プログラムモジュールを識別する識別番号とをそれぞれに付与する識別番号付与手段と、前記識別番号付与手段によって付与された前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、前記識別番号付与手段によって付与された前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するテーブル作成手段と、前記呼び出し元プログラムモジュールから前記他プログラムモジュールへの呼び出し命令を受けて、前記他プログラムモジュールの識別番号を取得する識別番号取得手段と、前記識別番号取得手段によって取得された識別番号をインデクスとして、前記テーブル作成手段によって作成されたテーブルから前記他プログラムモジュールのデータ領域の先頭アドレスを取得する先頭アドレス取得手段と、前記先頭アドレス取得手段によって取得された前記他プログラムモジュールのデータ領域の先頭アドレスと、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを切り替える先頭アドレス切替手段とをさらに備えることが好ましい。

【００８９】

この構成によれば、呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、他プログラムモジュールはコード領域とデータ領域とを含み、識別番号付与手段によって、呼び出し元プログラムモジュールを識別する識別番号と、他プログラムモジュールを識別する識別番号とがそれぞれに付与される。そして、テーブル作成手段によって、付与された呼び出し元プログラムモジュールの識別番号と、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、付与された他プログラムモジュールの識別番号と、他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルが作成される。識別番号取得手段によって、呼び出し元プログラムモジュールから他プログラムモジュールへの呼び出し命令を受けて、他プログラムモジュールの識別番号が取得される。先頭アドレス取得手段によって、取得された識別番号をインデクスとして、作成されたテーブルから他プログラムモジュールのデータ領域の先頭アドレスが取得される。先頭アドレス切替手段によって、取得された他プログラムモジュールのデータ領域の先頭アドレスと、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとが切り替えられる。

【００９０】

したがって、アドレス変換するためのテーブルを、ページ単位ではなくモジュール単位で行うことによって、テーブルのデータ量を削減することができ、資源制約の厳しい小型の機器に適合させることができる。

【００９１】

また、上記の目的プログラム実行装置において、前記呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、前記他プログラムモジュールはコード領域とデータ領域とを含み、前記呼び出し元プログラムモジュールを識別する識別番号と、前記他プログラムモジュールを識別する識別番号とをそれぞれに付与する識別番号付与手段と、前記識別番号付与手段によって付与された前記呼び出し元プログラムモジュールの識別番号と、前記呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、前記識別番号付与手段によって付与された前記他プログラムモジュールの識別番号と、前記他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するテーブル作成手段とをさらに備えることが好ましい。

【００９２】

この構成によれば、呼び出し元プログラムモジュールはコード領域とデータ領域とを含み、他プログラムモジュールはコード領域とデータ領域とを含み、識別番号付与手段によって、呼び出し元プログラムモジュールを識別する識別番号と、他プログラムモジュールを識別する識別番号とがそれぞれに付与される。そして、テーブル作成手段によって、付与された呼び出し元プログラムモジュールの識別番号と、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、付与された他プログラムモジュールの識別番号と、他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルが作成される。

【００９３】

したがって、アドレス変換するためのテーブルを、ページ単位ではなくモジュール単位で行うことによって、テーブルのデータ量を削減することができ、資源制約の厳しい小型の機器に適合させることができる。

【発明の効果】

【００９４】

本発明によれば、いわゆる機器組み込みプログラムのようにオブジェクトプログラムを格納することのできるメモリ領域が限られており、しかも近年の携帯情報機器のようにその限られたメモリ領域に高度且つ高機能な情報処理のためのオブジェクトプログラムを格納しなければならないような目的に適う、再入可能なオブジェクトプログラムを自動生成することができ、モジュール間の動的なリンクを実現することができる。

【発明を実施するための最良の形態】

【００９５】

本特許出願に係る発明（以後、「本発明」とも言う）の実施の形態について図を参照して説明する。

【００９６】

（第１の実施の形態）

図１は、本発明の一実施の形態による動的モジュールリンクシステムの構成を示す図である。図１に示す動的モジュールリンクシステム１０は、サーバ１１と複数の端末装置１２（１２ａ，１２ｂ，１２ｃ，・・・）とを備えて構成される。サーバ１１と各端末装置１２とは、例えば、インターネット１３を介して接続されている。なお、本実施の形態における端末装置とは、原始プログラムが変換されることによって生成される目的プログラムを実行する目的プログラム実行装置のことである。目的プログラム実行装置は、携帯電話機等の携帯通信端末や、インターネット１３等のネットワークに接続された家電機器等を含む。

【００９７】

なお、本実施の形態における端末装置は、インターネット１３等のネットワークを介してサーバ１１から目的プログラムを受信し、受信した目的プログラムを実行するものであるが、本発明は特にこれに限定されず、コンパイル装置によって生成された目的プログラムを、例えばＣＤ－ＲＯＭやＤＶＤ－ＲＯＭ等のコンピュータ読み取り可能な記録媒体に記録させ、端末装置が備える、例えばＣＤ－ＲＯＭドライブやＤＶＤ－ＲＯＭドライブ等の記録媒体駆動装置によって、前期記録媒体に記録された目的プログラムを読み込み、読み込まれた目的プログラムを実行してもよい。

【００９８】

サーバ１１は、コンパイル装置１４を備えて構成される。コンパイル装置１４は、プログラマによって作成された原始プログラムを目的プログラムに変換する。端末装置１２は、コンパイル装置１４によって変換された目的プログラムを実行する。

【００９９】

図２は、図１に示すコンパイル装置１４の構成を示すブロック図である。図２に示すように、コンパイル装置１４は、プログラム実行部１６及びプログラム記憶部１７を備えて構成される。

【０１００】

プログラム実行部１６は、例えばＣＰＵ（中央演算処理装置）等で構成され、アドレス保存プログラム生成部１６ａ、アドレス設定プログラム生成部１６ｂ、移行プログラム生成部１６ｃ、アドレス再設定プログラム生成部１６ｄ及びアクセスプログラム生成部１６ｅを備えて構成される。プログラム記憶部１７は、コンパイラプログラム記憶部１７ａ、ソースプログラム記憶部１７ｂ及びオブジェクトプログラム記憶部１７ｃを備えて構成される。

【０１０１】

アドレス保存プログラム生成部１６ａは、ソースプログラムに含まれる呼び出し元プログラムモジュールが使用する呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するためのアドレス保存プログラムを生成する。

【０１０２】

アドレス設定プログラム生成部１６ｂは、呼び出し元プログラムモジュールによって呼び出される他プログラムモジュールが使用する他プログラムモジュール用データメモリ領域アドレスを設定するためのアドレス設定プログラムを生成する。

【０１０３】

移行プログラム生成部１６ｃは、呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムへ移行するための移行プログラムを生成する。

【０１０４】

アドレス再設定プログラム生成部１６ｄは、移行された第２のサブプログラムから第１のサブプログラムへの復帰後に、保存された呼び出し元プログラムモジュール用データメ

モリ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成する。

【０１０５】

アクセスプログラム生成部１６ｅは、他プログラムモジュールが他プログラムモジュールに含まれる他プログラムモジュール用データメモリ領域をアクセスする際に、設定された他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域をアクセスするためのアクセスプログラムを生成する。

【０１０６】

コンパイラプログラム記憶部１７ａは、予め作成されたソースプログラム（原始プログラム）をオブジェクトプログラム（目的プログラム）に変換するためのコンパイラプログラムを記憶する。ソースプログラム記憶部１７ｂは、例えば、Ｃ言語やＣ＋＋言語等で記述されたソースプログラムを記憶する。なお、ソースプログラムは、アプリケーション全体ではなく、モジュール単位で作成される。オブジェクトプログラム記憶部１７ｃは、ソースプログラムから変換されたオブジェクトプログラムを記憶する。

【０１０７】

次に、図２に示すコンパイル装置によるコンパイラ処理について説明する。図３は、図２に示すコンパイル装置によるコンパイラ処理の一例を示すフローチャートである。

【０１０８】

ステップＳ１において、アドレス保存プログラム生成部１６ａは、アプリケーションプログラムモジュールのデータ領域の先頭アドレスをスタックに保存するためのアドレス保存プログラムを生成する。

【０１０９】

ステップＳ２において、アドレス設定プログラム生成部１６ｂは、ライブラリプログラムモジュールのデータ領域の先頭アドレスを設定するためのアドレス設定プログラムを生成する。

【０１１０】

ステップＳ３において、移行プログラム生成部１６ｃは、アプリケーションプログラムモジュールからライブラリプログラムモジュールに移行するための移行プログラムを生成する。

【０１１１】

ステップＳ４において、アドレス再設定プログラム生成部１６ｄは、スタックに保存したアプリケーションプログラムモジュールのデータ領域の先頭アドレスに再設定するためのアドレス再設定プログラムを生成する。

【０１１２】

ステップＳ５において、アクセスプログラム生成部１６ｅは、ライブラリプログラムモジュールがライブラリプログラムモジュールに含まれるライブラリプログラムモジュール用データ領域をアクセスする際に、設定されたライブラリプログラムモジュール用データ領域の先頭アドレスからの相対アドレスで、データ領域をアクセスするためのアクセスプログラムを生成する。

【０１１３】

このように、コンパイラプログラムが実行されることによって生成されるアドレス保存プログラム、アドレス設定プログラム、移行プログラム、アドレス再設定プログラム及びアクセスプログラムがオブジェクトプログラムとして各端末装置１２に送信され、端末装置１２によって、受信したオブジェクトプログラムが実行される。

【０１１４】

図４は、本発明の第１の実施の形態におけるコンパイル装置が変換するソースプログラムと、コンパイル装置によって変換されたオブジェクトプログラムとの構造を示す図であり、図４（ａ）は、アプリケーションプログラムモジュールの構造を示す図であり、図４（ｂ）は、ライブラリプログラムモジュールの構造を示す図である。

【０１１５】

但し、これらのプログラムは必ずしも１つにまとまった形で存在するとは限らず、幾つ

かの物理的な媒体、ファイル等に分散して記録されていることもある。

【０１１６】

これらのプログラムは、アプリケーションプログラムモジュール１２１とアプリケーションプログラムモジュール１３１とライブラリプログラムモジュール１５１と、その他本発明の説明に直接関係しないので図示していない幾つかのプログラムモジュールとを含んでいる。

【０１１７】

プログラムモジュールとは、プログラムを何らかの単位（例えば、機能単位、リンク単位、バージョンアップ単位等）で分割したものであり、プログラムコード領域と固有データ領域とを保持する。尚ここでは固有データ領域としたが、必ずしも厳密に固有であることは必要でなく、他の何らかの目的や用途と兼用しているものであっても良い。このことは他の説明についても同様である。

【０１１８】

プログラムモジュールのプログラムコード領域とは、そのプログラムモジュール内に記述されたプログラムの実行命令等を保持する領域である。

【０１１９】

プログラムモジュールの固有データ領域とは、そのプログラムモジュール内で定義された、外部変数、および静的変数の実体を保持する領域である。以後、外部変数と静的変数の両方を含めて、外部変数と言うこともある。

【０１２０】

アプリケーションプログラムモジュール１２１は、プログラムコード領域１２３とこのアプリケーションプログラムモジュール１２１の固有データ領域１２５と、その他本発明の説明に直接関係しないので図示していない幾つかの部分とを含んでいる。

【０１２１】

アプリケーションプログラムモジュール１３１は、プログラムコード領域１３３とこのアプリケーションプログラムモジュール１３１の固有データ領域１３５と、その他本発明の説明に直接関係しないので図示していない幾つかの部分とを含んでいる。

【０１２２】

これらのアプリケーションプログラムモジュール１２１とアプリケーションプログラムモジュール１３１とは、呼び出し元プログラムであって、共通にライブラリプログラムモジュール１５１を呼び出す。この呼び出しは通常、関数コール、サブルーチンコール等の形態で行われる。それが図４に示すプログラム呼び出し１とプログラム呼び出し２（再入：ｒｅｅｎｔｒｙ）である。

【０１２３】

これらプログラムモジュールは通常、幾つかのサブプログラムを含んでいる。このサブプログラムは通常、関数やサブルーチンやその他である。また、これら関数やサブルーチン１つをプログラムモジュールと呼ぶこともある。

【０１２４】

また通常、プログラムモジュールの呼び出しは、他プログラムモジュールのサブプログラムを呼び出すことによって行うが、同一プログラムモジュール内のサブプログラムを呼び出すこともある。

【０１２５】

いわゆるマルチタスク環境、その他時分割的に複数のプロセスまたはタスクまたはアプリケーションが並行して実行される、或いは実行しなければならない環境では、例えば、アプリケーションプログラムモジュール１２１がライブラリプログラムモジュール１５１を呼び出して実行中に、並行して、アプリケーションプログラムモジュール１３１がライブラリプログラムモジュール１５１を呼び出して実行することが発生する。

【０１２６】

このような状態は種々の情報機器で無数に近く発生するが、非常に簡単な幾つかの例を示す。

【０１２７】
例えば、携帯電話等のモバイル情報通信機能を有する機器では、ユーザーが送信するためのデータを入力している時にも、同時に並行して着信の有無を監視していなければならない。
【０１２８】
このデータ入力と着信との監視を別のＣＰＵで行うこともできるが、ＣＰＵの節約のため１つのＣＰＵで行おうとするならば、時分割的に並行して行う必要が起こる。
【０１２９】
そしてある基本的な関数型サブルーチンが、これらデータ入力アプリケーションと着信監視アプリケーションとの両方で使用される関数型サブルーチンであったならば、この関数型サブルーチンは、一方のアプリケーションから呼び出されて実行中に、もう一方のアプリケーションから呼び出されることが起こり得る。
【０１３０】
このように複数のタスク等が時間的に重複して同一のプログラムモジュール（関数またはサブプログラムまたはサブルーチン等）を呼び出し、実行するようなケースを「再入：ｒｅｅｎｔｒｙ」と言い、あるプログラムモジュールが時間的に重複・並行して複数のタスクを実行中のプログラムから呼び出され、実行され得るように構成されていることを「再入可能：ｒｅｅｎｔｒａｎｔ」と言う。
【０１３１】
従来から、プログラムモジュールを再入可能であるように構成する方法は幾つか知られている。
【０１３２】
その第１の方法は、いわゆる外部変数を使用せず、全て内部変数（以後、Ｌｏｃａｌ変数と言うこともあるが同一内容を表す）で処理することであった。内部変数はこのプログラムモジュールが呼び出される毎にスタック領域に新たに確保され、このプログラムモジュールの処理が終了するのと同時に解放されるため、全て内部変数を使用している限り、変数へのアクセスが衝突することは発生せず、同時に並行的に実行されても問題はなかった。
【０１３３】
その第２の方法は、いわゆる外部変数を使用するが、この外部変数に対するアクセスをいわゆるセマフォ（ｓｅｍａｐｈｏｒｅ：腕木信号機）と呼ばれる状態変数によって調停する方法である。
【０１３４】
第１の方法では、外部変数を使用することができないため、このプログラム以外のプログラムモジュールとデータとを受け渡しするのに制約を受けることがある。
【０１３５】
第２の方法では、煩雑なセマフォに対するテスト・アンド・セットを外部変数にアクセスする毎に行わなければならず、多くの負担をプログラマに課し、間違いや誤動作の原因となることが多かった。
【０１３６】
ライブラリプログラムモジュール１５１は、プログラムコード領域１５３とこのライブラリプログラムモジュール１５１の固有データ領域１５５、１５７と、その他本発明の説明に直接関係しないので図示していない幾つかの部分とを含んでいる。
【０１３７】
アプリケーションプログラムモジュール１２１用データ領域１５５とアプリケーションプログラムモジュール１３１用データ領域１５７とは、このライブラリプログラムモジュール１５１を呼び出した元のプログラムモジュールによって区別されている。
【０１３８】
例えば、アプリケーションプログラムモジュール１２１からこのライブラリプログラムモジュール１５１が呼び出された時には、プログラムコード領域１５３は外部変数領域と

してアプリケーションプログラムモジュール１２１用データ領域１５５をアクセスしてその処理を行い、アプリケーションプログラムモジュール１３１からこのライブラリプログラムモジュール１５１が呼び出された時には、プログラムコード領域１５３は外部変数領域としてアプリケーションプログラムモジュール１３１用データ領域１５７をアクセスしてその処理を行う。

【０１３９】

このように呼び出し元プログラム単位に、ライブラリプログラムモジュールが使用する固有データ領域、特に外部変数のために使用するデータ領域を区分して使用することによって、外部変数に対するアクセスが衝突することを気にせず、プログラムを作成することができる。

【０１４０】

なお、既に説明したように、内部変数については、このライブラリプログラムモジュール１５１が呼び出される毎に、領域がスタックに確保されるため、アクセスが衝突する問題は生じない。

【０１４１】

尚ここで、データ領域を呼び出し元アプリケーションプログラム単位に用意して使用すると説明したが、ここで言うアプリケーションプログラムを、アプリケーションタスク単位と読み替えても良く、ここで言うアプリケーションプログラムはアプリケーションタスクと同じ概念であると考えても良い。

【０１４２】

また、ここで言うアプリケーションプログラムを、アプリケーションプロセス単位と読み替えても良く、ここで言うアプリケーションプログラムはアプリケーションプロセスと同じ概念であると考えても良い。

【０１４３】

より詳しくアプリケーションプログラムモジュール１２１がライブラリプログラムモジュール１５１を呼び出す時の処理手順について図５～図８を用いて説明する。

【０１４４】

図５は、図１に示す端末装置１２の構成を示すブロック図である。図６は、図５に示す端末装置１２によってアプリケーションプログラムモジュール１２１がライブラリプログラムモジュール１５１を呼び出す処理を示すフローチャートである。図７は、アプリケーションプログラムモジュールからライブラリプログラムモジュールを呼び出す処理を説明するための図であり、図７（ａ）は、アプリケーションプログラムモジュールの処理を説明するための図であり、図７（ｂ）は、ライブラリプログラムモジュールの処理を説明するための図である。図８は、アプリケーションプログラムモジュール及びライブラリプログラムモジュールのソースプログラム及びオブジェクトプログラムの一例を示す図であり、図８（ａ）は、アプリケーションプログラムモジュールのソースプログラム及びオブジェクトプログラムの一例を示す図であり、図８（ｂ）は、ライブラリプログラムモジュールのソースプログラム及びオブジェクトプログラムの一例を示す図である。なお、図８（ａ）、（ｂ）に示すソースプログラムは、Ｃ言語で記述されたプログラムを例として記述している。

【０１４５】

図５に示すように、端末装置１２は、プログラム実行部１８及びメモリ１９を備えて構成される。プログラム実行部１８は、例えばＤＳＰ（Digital Signal Processor）で構成され、アドレス保存部１８ａ、アドレス設定部１８ｂ、移行部１８ｃ、アドレス再設定部１８ｄ及びアクセス部１８ｅを備えて構成される。メモリ１９は、アプリケーションプログラムモジュール記憶部１９ａ、ライブラリプログラムモジュール記憶部１９ｂ及びスタック１９ｃを備えて構成される。

【０１４６】

アプリケーションプログラムモジュール記憶部１９ａは、コンパイル装置１４から送信されたアプリケーションプログラムモジュールを記憶する。ライブラリプログラムモジュ

ール記憶部１９ｂは、コンパイル装置１４から送信されたライブラリプログラムモジュールを記憶する。スタック１９ｃは、サブルーチンや関数を呼び出す際の処理中のデータやアドレス等を一時的に記憶する。

【０１４７】

アドレス保存部１８ａは、呼び出し元プログラムモジュールであるアプリケーションプログラムモジュール１２１が使用するアプリケーションプログラムモジュール用固有データ領域１２５の先頭アドレスをスタック１９ｃに保存する。

【０１４８】

アドレス設定部１８ｂは、アプリケーションプログラムモジュール１２１によって呼び出されるライブラリプログラムモジュール１５１のアプリケーションプログラムモジュール用データ領域１５５の先頭アドレスを設定する。

【０１４９】

移行部１８ｃは、アプリケーションプログラムモジュール１２１に含まれるプログラムコード領域１２３からライブラリプログラムモジュール１５１に含まれるプログラムコード領域１５３へ移行する。

【０１５０】

アドレス再設定部１８ｄは、移行されたライブラリプログラムモジュール１５１のプログラムコード領域１５３からアプリケーションプログラムモジュール１２１のプログラムコード領域１２３への復帰後に、保存されたアプリケーションプログラムモジュール用データ領域１２５の先頭アドレスを読み出して再設定する。

【０１５１】

アクセス部１８ｅは、ライブラリプログラムモジュール１５１のプログラムコード領域１５３がライブラリプログラムモジュール１５１に含まれるライブラリプログラムモジュール用データ領域１５５をアクセスする際に、設定されたライブラリプログラムモジュール用データ領域１５５の先頭アドレスからの相対アドレスで、データ領域１５５をアクセスする。

【０１５２】

ここで、アプリケーションプログラムモジュール１２１のソースプログラムに記述された「ｂａｒ（Ａ，Ｂ）；」命令がオブジェクトプログラムに変換されると、ライブラリプログラムモジュール１５１に含まれる関数型サブルーチンであるｂａｒ（以後、関数型サブプログラムｂａｒと言うこともあり、単に関数ｂａｒと言うこともあるが同一である）を呼び出すオブジェクトプログラムが生成される（図７（ａ）の１～７）。

【０１５３】

図６のステップＳ１１において、アドレス保存部１８ａは、現在のＤＰ（データポインタ）レジスタ（以後単にＤＰと言うこともある）の内容をスタック１９ｃに保存する。アプリケーションプログラムモジュール１２１のプログラムコード領域１２３に含まれるプログラムは、ライブラリプログラムモジュール１５１に含まれる関数ｂａｒを呼び出すに際し、まず現在実行中のアプリケーションプログラムモジュール１２１が使用している固有データ領域１２５の先頭アドレスである０ｘ０６００（１６進数表記で０６００であり、１０進数表記に変換すると、$0*16^3+6*16^2+0*16+0$である、以後同じ）を保存しているＤＰレジスタの内容をスタックに保存する（図７（ａ）の１、図８（ａ）の１））。

【０１５４】

これはライブラリプログラムモジュール１５１から戻ってきた時に、アプリケーションプログラムモジュール１２１がＤＰの内容を再度０ｘ０６００に設定し、アプリケーションプログラムモジュール１２１が自分自身のための固有データ領域１２５を再度使用することができるようにするためである。尚ここでは固有のデータ領域としたが、必ずしも厳密に固有であることは必ずしも必要でなく、他の何らかの用途や目的、その他と共有であっても良い。このことは他の説明についても同様である。

【０１５５】

ＤＰレジスタは、実行中のプログラムモジュールのデータ領域の先頭アドレスを保持する領域である。この領域として、通常特別なレジスタを用意するが、必ずしも特別なレジスタを用意せず、汎用的なレジスタを使用してもよい。

【０１５６】

次に、ステップＳ１２において、アドレス保存部１８ａは、関数ｂａｒの引数をスタック１９ｃに保存する。すなわち、アプリケーションプログラムモジュール１２１に含まれるプログラムは、ライブラリプログラムモジュール１５１に引き渡すデータである引数Ａと引数Ｂとをスタック１９ｃに保存する（図７（ａ）の２、図８（ａ）の２）。

【０１５７】

ここで、この時のスタック１９ｃの状態について説明する。図９は、スタックの状態を説明するための図であり、図９（ａ）は、引数Ａと引数Ｂとをスタックに保存した際のスタックの状態を示す概念図であり、図９（ｂ）は、スタックの使われ方を説明するための概念図である。

【０１５８】

図９（ａ）に示すようにスタック（以後、スタックメモリと言うこともある）１９ｃは Ｌａｓｔ－ｉｎ／Ｆｉｒｓｔ－ｏｕｔ型のメモリ領域で、あるプログラムモジュールから他プログラムモジュール（例えば、関数やサブルーチン、サブプログラム、その他である）を呼び出す時に必要なデータを保存して、プログラムモジュール間でデータの引き渡しに使ったり、Ｌｏｃａｌ変数用メモリ領域として使用したり、呼び出された関数等から呼び出し元のプログラムに戻るためのアドレスの保存や、その他の目的に使用されるものである。

【０１５９】

スタック１９ｃのデータを出し入れする領域のアドレスは、スタックポインタと呼ばれるレジスタで示される（以後、単にスタックポインタ或いはスタックポインタレジスタと呼ぶこともある）。

【０１６０】

このスタックの使われ方を図９（ｂ）を用いて説明する。例えば、図９（ｂ）に示すスタックの上から３番目に格納されているローカル変数２に対してアクセスするには、スタックポインタの値を変更するか或いは、スタックポインタレジスタの値マイナス２の値をアドレスとして間接参照を行うこともできる。

【０１６１】

次に、ステップＳ１３において、アドレス設定部１８ｂは、新しいＤＰを設定する。すなわち、アプリケーションプログラムモジュール１２１に含まれるプログラムは、自分自身の固有データ領域１２５内の領域に記憶している、関数ｂａｒを含むライブラリプログラムモジュール１５１で使用する、アプリケーションプログラムモジュール１２１用データ領域１５５のアドレス（０ｘ８００）を読み出し、そのアドレスをＤＰに設定する（図７（ａ）の３、図８（ａ）の３）。

【０１６２】

前記関数ｂａｒを含むライブラリプログラムモジュール１５１で使用する、アプリケーションプログラムモジュール１２１用データ領域１５５のアドレス（０ｘ８００）を保持する領域は、通常コンパイラが設定する。しかし必ずしも、前記領域をコンパイラが設定せず、ライブラリとして用意し、リンク時に設定してもよい。

【０１６３】

次に、ステップＳ１４において、移行部１８ｃは、ライブラリ関数を呼び出す。アプリケーションプログラムモジュール１２１に含まれるプログラムは、呼び出し先プログラムである関数ｂａｒから自分自身に帰ってくる時のアドレス、即ち関数ｂａｒへのジャンプ命令の次の命令のアドレスをスタック１９ｃに保存し、関数ｂａｒへジャンプする（図７（ａ）の４、図８（ａ）の４）。

【０１６４】

図７（ｂ）に示すように、呼び出されるプログラムである関数ｂａｒは、ライブラリプ

ログラムモジュール１５１に含まれるプログラムコード領域１５３に含まれている。この関数ｂａｒは引数としてスタック１９ｃに保存された引数Ａと引数Ｂとを取り出して処理に使用する。この時のスタック１９ｃの状態は前記図９（ａ）に示す通りである。

【０１６５】

更に、ステップＳ１５において、アクセス部１８ｅは、外部変数に値を代入する。関数ｂａｒは外部変数として、例えばこのライブラリプログラムモジュール１５１で定義された外部変数Ｃを使用する。関数ｂａｒではこの外部変数Ｃに対してアクセスし、例えばそのメモリ領域に値１０を代入するに際し、ＤＰに保存されている値（０ｘ８００）にＤＰからの相対アドレス（０ｘ５０）を加算したアドレス（０ｘ８５０）に値１０を代入する。この命令形式を図７（ｂ）の５と図８（ｂ）の５に示す。

【０１６６】

これによって関数ｂａｒは、アドレス０ｘ８００から始まるメモリ領域であるアプリケーションプログラムモジュール１２１用データ領域１５５に対してアクセスすることができる。

【０１６７】

関数ｂａｒでの処理が終ると、プログラムの制御は元のアプリケーションプログラムモジュール１２１に含まれるプログラムに戻される。

【０１６８】

そして、ステップＳ１６において、アドレス再設定部１８ｄは、関数ｂａｒの引数をスタック１９ｃから破棄する。関数ｂａｒからの復帰直後に、スタック１９ｃから引数Ａ、Ｂが破棄される（図７（ａ）の６、図８（ａ）の６）。尚ここでは関数ｂａｒからの復帰直後としたが、必ずしも厳密に直後であることは必要でなく、何らかの処理の後であっても良い。このことは他の説明についても同様である。

【０１６９】

次に、ステップＳ１７において、アドレス再設定部１８ｄは、スタック１９ｃに保存していた元のＤＰの値、この例では０ｘ６００がスタック１９ｃから取り出されてＤＰに戻される（図７（ａ）の７、図８（ａ）の７）。

【０１７０】

これによって呼び出し元のアプリケーションプログラムモジュール１２１に含まれるプログラムは、再度自分自身の固有データ領域１２５に対してアクセスすることが可能になる。

【０１７１】

尚この時、図９（ａ），（ｂ）に示す通り、呼び出し元のＤＰの値は、スタックの一番下に保存されているので、呼び出されたライブラリプログラムモジュール１５１の側ではその内容を読み出す必要がないと共に、実際に読み出さなくても他の必要な処理を全て行うことが可能であり、不要なアクセスを行う必要が生じない。

【０１７２】

また、例えば次の第２の実施の形態に示すような他の実施の形態であって、ＤＰをスタックに保存するか保存しないか選択ができる実施の形態では、ＤＰをスタックの一番下に保存することによって、ＤＰがスタックに保存されているか保存されていないかに係わらず、同一アドレスに対するアクセスによって同じ処理を行うことができ、ＤＰがスタックに保存されているか否かを判断して処理を分ける必要がない。

【０１７３】

ＤＰをスタックの一番下に保存することによって、ＤＰがスタックに保存されているか保存されていないかに係わらず、呼び出されたライブラリプログラムモジュール１５１の側で必要なデータ、例えば引数や帰りアドレスを取り出すスタック位置は変わらない。すなわち、引数や帰りアドレスを取り出すスタックのスタックポインタからの相対位置は変化しないので、それを考慮してこれら引数や帰りアドレスを取り出すことは必要でない。

【０１７４】

尚、この例では、アプリケーションプログラムモジュールから、ライブラリプログラム

モジュールを呼び出す例について書いたが、ライブラリプログラムモジュールから、他のライブラリモジュールを呼び出すことも可能であり、その際の処理は、アプリケーションプログラムモジュールから、他ライブラリモジュールを呼び出す場合の処理と同じであるから、説明を省略する。

【０１７５】

尚、この例では、他プログラムモジュールを呼び出す例について書いたが、同一プログラムモジュールを呼び出すことも可能であり、その際の処理は、他プログラムモジュールを呼び出す場合の処理と同じであるから、説明を省略する。

【０１７６】

このように、他プログラムモジュールに対して呼び出し元プログラムモジュールによる再入が発生したとしても、アプリケーションプログラムモジュール１２１が使用する固有データ領域１２５の先頭アドレスがスタックに保存されるので、アプリケーションプログラムモジュール１２１が使用する固有データ領域１２５の先頭アドレスを退避させることができ、アプリケーションプログラムモジュール１２１によって呼び出されるライブラリプログラムモジュールが使用するアプリケーションプログラムモジュール１２１用データ領域１５５の先頭アドレスが設定されるので、再入可能なオブジェクトプログラムを生成することができ、モジュール間の動的リンクを実現することができる。

【０１７７】

（第２の実施の形態）

前記本発明の第１の実施の形態では、アプリケーションプログラムモジュール１２１のソースプログラムに記述された「ｂａｒ（Ａ，Ｂ）；」命令がオブジェクトプログラムに変換されることによって生成されるオブジェクトプログラムは、現在実行中のアプリケーションプログラムモジュール１２１が使用している固有データ領域１２５の先頭アドレスを保存しているＤＰレジスタの内容をスタックに保存し、ライブラリプログラムモジュール１５１に引き渡すデータである引数Ａと引数Ｂをスタックに保存し、呼び出される関数ｂａｒで使用する、アプリケーションプログラムモジュール１２１用データ領域１５５のアドレスを読み出してそのアドレスをＤＰに設定し、関数ｂａｒから自分自身に帰ってくる時のアドレスをスタックに保存し、関数ｂａｒへジャンプし、関数ｂａｒからの復帰時には、引数Ａと引数Ｂをスタックから破棄し、スタックに保存している元のＤＰの値を再設定するものであった。

【０１７８】

次に説明する本発明の第２の実施の形態では、上記のような処理を行うか、それとも上記の処理の一部を行わずに省略するかの記述を、ライブラリプログラムモジュール１５１のソースプログラムが含むものである。

【０１７９】

なお、上述の第１の実施の形態では、他プログラムモジュールの呼び出しを例に挙げ説明したが、本発明の第２の実施の形態では、同一モジュール内の関数ｂａｒの呼び出しを例に挙げて説明する。

【０１８０】

図１０は、第２の実施の形態におけるソースプログラム及びオブジェクトプログラムの一例を示す図であり、図１０（ａ）は、ライブラリプログラムモジュールのソースプログラムの一例を示す図であり、図１０（ｂ）は、ライブラリプログラムモジュールのオブジェクトプログラムの一例を示す図である。なお、図１０ではＣ言語を例に記述している。この例ではＣ言語の言語仕様を拡張して、新規キーワードとして「ｐｒｉｖａｔｅ」を追加し、ソースコード上で「ｐｒｉｖａｔｅ」修飾詞をつけることにより、宣言を記述している。

【０１８１】

例えば、図１０に示す関数ｂａｒの呼び出しでは、上記本発明の第１の実施の形態で行った処理の一部を行うことなく、省略できることを示す。図１０（ａ）に示す関数ｂａｒでは、処理の一部を省略するための宣言が、例えば、「ｐｒｉｖａｔｅ　ｖｏｉｄ　ｂａ

ｒ（ｉｎｔ　Ａ，ｉｎｔ　Ｂ）」で記述されている。

【０１８２】

図１０（ａ）に示すように、ライブラリモジュールプログラムのソースプログラムに含まれる関数ｂａｒには、関数ｂａｒが「ｐｒｉｖａｔｅ」である旨の記述が含まれている。

【０１８３】

この記述は「関数ｂａｒがいわゆるプライベイト（ｐｒｉｖａｔｅ）関数である」旨を宣言する記述であり、例えば「プライベイト（ｐｒｉｖａｔｅ）宣言」或いは「プライベイト（ｐｒｉｖａｔｅ）関数宣言」と呼ばれることもある。

【０１８４】

このプライベイト関数ｂａｒを呼び出す記述がある場合、第１の実施の形態で説明したような、呼び出し元であるプログラムモジュールが使用しているＤＰの保存や、呼び出されるプログラムモジュールで使用される固有データ領域アドレスを示すＤＰの設定や、関数ｂａｒからの復帰時に、スタックに保存している元のＤＰの値の再設定を行うためのオブジェクトプログラムは生成されない。

【０１８５】

図１１は、本発明の第２の実施の形態におけるコンパイル装置の構成を示すブロック図である。図１１に示すコンパイル装置１４は、プログラム実行部１６及びプログラム記憶部１７を備えて構成される。

【０１８６】

プログラム実行部１６は、例えばＣＰＵ（中央演算処理装置）等で構成され、アドレス保存プログラム生成部１６ａ、アドレス設定プログラム生成部１６ｂ、移行プログラム生成部１６ｃ、アドレス再設定プログラム生成部１６ｄ、アクセスプログラム生成部１６ｅ及び宣言判別部１６ｆを備えて構成される。プログラム記憶部１７は、コンパイラプログラム記憶部１７ａ、ソースプログラム記憶部１７ｂ及びオブジェクトプログラム記憶部１７ｃを備えて構成される。

【０１８７】

なお、図１１に示す第２の実施の形態におけるコンパイル装置と、図２に示す第１の実施の形態におけるコンパイル装置とで同じ構成物については、同じ符号を付し、以下の説明では第１の実施の形態とは異なる構成についてのみ説明する。

【０１８８】

宣言判別部１６ｆは、ソースプログラムに含まれるアプリケーションプログラムモジュールに含まれるプログラムコード領域から呼び出されるライブラリプログラムモジュールのプログラムコード領域の記述に基づいて、処理を短縮するか否かを判別する。すなわち、宣言判別部１６ｆは、ソースプログラムの記述に上述のプライベイト宣言が含まれているか否かを判別する。

【０１８９】

ここで、宣言判別部１６ｆによって処理を短縮しないと判別された場合、すなわち、プライベイト宣言が含まれていないと判別された場合、第１の実施の形態と同様に、アドレス保存プログラム生成部１６ａ、アドレス設定プログラム生成部１６ｂ、移行プログラム生成部１６ｃ、アドレス再設定プログラム生成部１６ｄ及びアクセスプログラム生成部１６ｅによる処理が行われる。

【０１９０】

また、宣言判別部１６ｆによって処理を短縮しないと判別された場合、すなわち、プライベイト宣言が含まれていると判別された場合、移行プログラム生成部１６ｃは、アプリケーションプログラムモジュールに含まれるプログラムコード領域からライブラリプログラムモジュールに含まれるプログラムコード領域へ移行するための移行プログラムを生成し、アクセスプログラム生成部１６ｅは、ライブラリプログラムモジュールに含まれるプログラムコード領域がライブラリプログラムモジュールに含まれるライブラリプログラムモジュール用データ領域をアクセスする際に、ライブラリプログラムモジュール用データ

領域の先頭アドレスからの相対アドレスで、データ領域をアクセスするためのアクセスプログラムを生成する。

【０１９１】

図１２は、図１１に示すコンパイル装置によるコンパイラ処理の一例を示すフローチャートである。

【０１９２】

ステップＳ２１において、宣言判別部１６ｆは、ライブラリプログラムモジュールのプログラムコード領域の記述にプライベイト宣言が含まれているか否かを判断する。ここで、プライベイト宣言が含まれていないと判断されると（ステップＳ２１でＮＯ）、ステップＳ２２に移行し、プライベイト宣言が含まれていると判断されると（ステップＳ２１でＹＥＳ）、ステップＳ２７に移行する。

【０１９３】

ステップＳ２２からステップＳ２６までの処理は、図３に示すステップＳ１からステップＳ５までの処理と同じであるので説明を省略する。また、ステップＳ２７における処理も、図３に示すステップＳ３における処理と同じであるので説明を省略する。

【０１９４】

ステップＳ２８において、アクセスプログラム生成部１６ｅは、ライブラリプログラムモジュールに含まれるプログラムコード領域がライブラリプログラムモジュールに含まれるライブラリプログラムモジュール用データ領域をアクセスする際に、ライブラリプログラムモジュール用データ領域アドレスからの相対アドレスで、データ領域をアクセスするためのアクセスプログラムを生成する。

【０１９５】

図１３は、本発明の第２の実施の形態におけるライブラリプログラムモジュールのコード領域及びデータ領域を示す図である。このプライベイト関数ｂａｒを呼び出す記述がライブラリプログラムモジュール１５１のプログラムコード領域１５３で行われると、図１１に示すように、関数ｂａｒで使用するためのデータである引数Ａや引数Ｂを関数ｂａｒに引き渡すために、当該引数Ａ及び引数Ｂをスタックへ保存するステップと、関数ｂａｒへジャンプする命令の次の命令のアドレス（戻り番地）をスタックへ格納し、関数ｂａｒへジャンプするステップと、関数ｂａｒからの復帰時に、引数Ａや引数Ｂをスタックから破棄するステップとが実行されるオブジェクトプログラムが本実施の形態のコンパイル装置によって生成される。

【０１９６】

これによって、必ずしも呼び出し元であるプログラムモジュールと呼び出されるプログラムモジュールとが異なるデータ領域を使用することが必要でないケースでは、関数ｂａｒをプライベイト宣言することによって、ＤＰの保存や設定や再設定のステップを省略し、オブジェクトプログラムの実行を高速化することができ、不要なステップを出力しないことでオブジェクトプログラムサイズを小さくすることもできる。

【０１９７】

このようなプライベイト宣言された関数ｂａｒが実行されると、図１１に示すように、ＤＰの内容は呼び出し元プログラムであるプログラムモジュールの状態のまま変化していないので、例えば外部変数Ｃに対してデータの読み出し等が実行されると、呼び出し元であるプログラムモジュールの固有データ領域（この場合、固有データ領域１５５）がアクセスされる。

【０１９８】

また、図示していないが、プライベイト宣言された関数ｂａｒから呼び出し元プログラムへリターンした時にも、ＤＰの再設定等を行う必要はない。

【０１９９】

またこれまで説明した実施の形態では、例えば関数ｂａｒの前に「ｐｒｉｖａｔｅ」と記述されるとプライベイト関数であることが宣言され、ＤＰの保存等の一連の処理が行われず、特に何も記述されなければプライベイト関数でなく、ＤＰの保存等の処理が行われ

る。

【０２００】

しかし他の実施の形態では、例えば明示的にプライベイト関数でないことを示すために「ｐｕｂｌｉｃ」等の記述を行うことも可能である。

【０２０１】

このような記述は、「パブリック（ｐｕｂｌｉｃ）宣言」或いは「パブリック（ｐｕｂｌｉｃ）関数宣言」と呼ばれることもあり、この記述を行うことによって明示的にＤＰの保存等の一連の処理を行うことを示すことができる。

【０２０２】

また他の実施の形態では、例えばこの「プライベイト宣言」が行われる関数は、同一プログラムモジュール内でのみ呼び出される、或いは呼び出すことが可能な関数であっても良く、更に他の実施の形態では、この「プライベイト宣言」が行われる関数は、異なるプログラムモジュール間でのみ呼び出される、或いは呼び出すことが可能な関数であっても良い。

【０２０３】

尚、この例では新規キーワード「ｐｒｉｖａｔｅ」や「ｐｕｂｌｉｃ」を追加し、Ｃ言語を拡張して宣言を行っているが、プラグマ等を用いることにより、言語使用を拡張することなく、宣言を行ってもよい。また、別ファイルとして記述したり、コンパイラの引数として指定してもよい。

【０２０４】

このように、アプリケーションプログラムモジュールに含まれるプログラムコード領域から呼び出されるライブラリプログラムモジュールのプログラムコード領域の記述に基づいて、処理を短縮するか否かが判別されるので、必ずしもアプリケーションプログラムモジュールとライブラリプログラムモジュールとが異なるデータ領域を使用することが必要でないケースでは、アドレスの保存や設定や再設定の処理を省略し、オブジェクトプログラムの実行を高速化することができ、不要な処理を出力しないことでオブジェクトプログラムのサイズを小さくすることができる。

【０２０５】

また、ライブラリプログラムモジュールのプログラムコード領域の記述が、当該プログラムコード領域毎に記述される宣言であるので、この宣言の有無により、ライブラリプログラムモジュールのプログラムコード領域の処理内容に応じてアドレスの保存や設定や再設定の処理を省略することができ、オブジェクトプログラムの実行を高速化することができる。

【０２０６】

（第３の実施の形態）

前記本発明の第１の実施の形態では、関数ｂａｒが使用する全ての外部変数（例えばＣ）に対してアクセスするに際し、ＤＰに保存されている値（例えば、０ｘ８００）に、ＤＰからの相対アドレス（例えば、０ｘ５０）を加算したアドレス（例えば、０ｘ８５０）でアクセスしている。

【０２０７】

その結果、関数ｂａｒは、使用する全ての外部変数用領域として、アドレス例えば、０ｘ８００から始まるメモリ領域であるアプリケーションプログラムモジュール１２１用固有データ領域１５５を使用することができる。

【０２０８】

しかし次に説明する本発明の第３の実施の形態では、例えば関数ｂａｒを呼び出す複数のアプリケーションプログラムモジュールが共通にアクセスするデータメモリ領域に関する記述を、例えば関数ｂａｒの原始プログラムはその変数宣言部に含むことができる。

【０２０９】

図１４は、本発明の第３の実施の形態におけるライブラリプログラムモジュールのソースプログラム及びオブジェクトプログラムの一例を示す図であり、図１４（ａ）は、第３

の実施の形態におけるライブラリプログラムモジュールのソースプログラムの一例を示す図であり、図１４（ｂ）は、第３の実施の形態におけるライブラリプログラムモジュールのオブジェクトプログラムの一例を示す図である。

【０２１０】

図１４では、Ｃ言語を例として宣言を記述している。この例ではＣ言語の言語仕様を拡張して新規のキーワード「ｃｏｍｍｏｎ」を追加し、ソースコード上でこの「ｃｏｍｍｏｎ」修飾詞をつけることにより宣言を記述している。

【０２１１】

例えば、図１４（ａ）に示す関数ｂａｒのプログラムでは、関数ｂａｒが使用する外部変数としてＤとＣを使用する。その宣言が例えば図１４（ａ）の「ｃｏｍｍｏｎ　ｉｎｔ　Ｄ；」と「ｉｎｔ　Ｃ；」の記述である。

【０２１２】

そして更に「ｃｏｍｍｏｎ　ｉｎｔ　Ｄ；」の記述は、この外部変数Ｄはこの関数ｂａｒの複数の呼び出し元アプリケーションプログラムが共通にアクセスする外部変数であることを宣言している。

【０２１３】

尚、この例では新規のキーワード「ｃｏｍｍｏｎ」を追加し、Ｃ言語を拡張して宣言を行っているが、プラグマ等を用いて言語仕様を拡張することなく宣言を行ってもよい。

【０２１４】

また、別ファイル等に記述したり、コンパイラの引数として指定することによって宣言を行ってもよい。

【０２１５】

図１４（ａ）に示すように、ライブラリモジュールプログラムのソースプログラムに含まれる外部変数Ｄには、この外部変数Ｄが、「ｃｏｍｍｏｎ」である旨の記述が含まれている。この記述は、「外部変数Ｄがいわゆる共通外部変数である」旨を宣言する記述であり、例えば「コモン（ｃｏｍｍｏｎ）宣言」或いは「コモン（ｃｏｍｍｏｎ）外部変数宣言」と呼ばれることもある。

【０２１６】

図１５は、本発明の第３の実施の形態におけるライブラリプログラムモジュールのコード領域及びデータ領域を示す図である。

【０２１７】

図１５（ａ）は、プログラムコード領域５５３が、再入可能な関数ｂａｒに相当し、このプログラムから固有データ領域であるアプリケーションプログラムモジュール１２１用データ領域５５５と、共有データ領域である共有データ領域５５９との両方がアクセスされ得ることを示している。尚ここでは共有データ領域としたが、必ずしも厳密に他の何らかの目的や用途と共有されていることは必要でなく、たまたま他の何らかの目的や用途と共有されていないものであってもかまわない。このことは他の説明についても同様である。

【０２１８】

図１５（ｂ）は、より具体的に、外部変数Ｃは固有データ領域であるアプリケーションプログラムモジュール１２１用データ領域５５５に格納され、外部変数Ｄは共有データ領域アクセスによってアクセスされる共有データ領域５５９に格納されることを示している。

【０２１９】

図１４（ａ）の「ｉｎｔ　Ｃ；」の記述は、この外部変数Ｃは、この関数ｂａｒの複数の呼び出し元アプリケーションプログラム毎に固有のメモリ領域を使用することを宣言している記述である。この記述によって、図１５（ｂ）に示すように、外部変数Ｃは固有データ領域アクセスによってアクセスされるアプリケーションプログラムモジュール（この例ではプログラムモジュール１２１）用の固有のデータ領域５５５に格納される。

【０２２０】

そして関数ｂａｒでは、呼び出し元アプリケーションプログラム（例えばプログラムモジュール１２１）毎に固有のデータ領域５５５に格納されている外部変数Ｃに対してアクセスし、例えばそのメモリ領域に、値１０を代入するに際し、ＤＰに保存されている値（例えば、０ｘ８００）に、ＤＰからの相対アドレス（例えば、０ｘ５０）を加算したアドレス（例えば、０ｘ８５０）に、値１０を代入する。この命令形式を図１４（ｂ）と図１５（ｂ）とに示す。この時の命令形式は前記第１の実施の形態と同一である。

【０２２１】

図１５（ｂ）に示すように、関数ｂａｒは、例えばアドレス０ｘ８００から始まるメモリ領域であるアプリケーションプログラムモジュール１２１用の固有データ領域５５５に対してアクセスすることができる。また、関数ｂａｒに複数のアプリケーションプログラムモジュール（例えば、図４（ａ）のアプリケーションプログラムモジュール１２１とアプリケーションプログラムモジュール１３１）から同時に呼び出され、その結果、再入が起こっても外部変数へのアクセスが衝突することはない。

【０２２２】

図１４（ａ）の「ｃｏｍｍｏｎ　ｉｎｔ　Ｄ；」の記述は、この外部変数Ｄは、関数ｂａｒの複数の呼び出し元アプリケーションプログラムが共有するメモリ領域を使用することを宣言している記述である。この記述によって、図１５（ｂ）に示すように、外部変数Ｄは共有データ領域アクセスによってアクセスされる複数のアプリケーションプログラムモジュールが共通に使用する共有データ領域５５９に格納される。

【０２２３】

そして関数ｂａｒでは、呼び出し元アプリケーションプログラム（例えばプログラムモジュール１２１）が共通に使用する共有データ領域５５９に格納されている外部変数Ｄに対してアクセスし、例えばそのメモリ領域に、値２０を代入するに際し、特別なレジスタＳＤＰに保存されている値（例えば、０ｘＦ００）に、ＳＤＰからの相対アドレス（例えば、０ｘ６０）を加算したアドレス（例えば、０ｘＦ６０）に、値２０を代入する。この命令形式を図１４（ｂ）と図１５（ｂ）とに示す。

【０２２４】

このＳＤＰは特別なレジスタであり、全アプリケーションプログラムに対して１つの共有データ領域の先頭アドレスを保存している。またこのＳＤＰに保存されている値である全アプリケーションプログラムに対して１つの共有データ領域の先頭アドレス自体をＳＤＰと呼ぶこともある。

【０２２５】

或いは他の実施の形態ではこのＳＤＰとして必ずしも特別なレジスタを使用せず、汎用的なレジスタを使用することもできる。このような実施の形態は汎用のレジスタを多く備えているようなＭＰＵ（Micro Processing Unit）において特に有効であり、更にこの汎用レジスタがメモリアクセスのためのオフセットアドレスとして使用できるＭＰＵでは更に有効である。

【０２２６】

或いはまた他の実施の形態ではこのＳＤＰとして必ずしも特別なレジスタや汎用のレジスタを使用せず、通常のメモリやスタックを使用することもできる。

【０２２７】

しかしこのような実施の形態ではこのスタックや通常のメモリに格納されたＳＤＰをオフセットアドレスとして使用してメモリアクセスするために、幾らかの処理が必要になることもある。

【０２２８】

例えばその処理とは、スタックやメモリに格納されているＳＤＰを読み出し、メモリアクセスのためのオフセットアドレスとして使用できるレジスタに格納したり、そのための準備として現在のレジスタの内容を退避する処理等である。

【０２２９】

これらのような方法で、このＳＤＰレジスタが記憶している値をオフセットアドレスと

して使用してメモリ領域アクセスすることによって、図７（ｂ）に示すように、関数ｂａｒは、例えばアドレス０ｘＦ００から始まるメモリ領域であるアプリケーションプログラムモジュール共有データ領域５５９に対してアクセスすることができる。

【０２３０】

ＳＤＰの値は図示していないが、例えば図７（ａ）に示すように、ライブラリプログラムモジュール１５１の自分固有データ領域アドレスと同様に固有データ領域１２５に記憶しておき、アプリケーションプログラムモジュール１２１からライブラリプログラムモジュール１５１を呼び出す時、その呼び出す処理の中で保存と設定を行うこともできる。

【０２３１】

例えば、図３のステップＳ１で現在のＤＰ（０ｘ０６００）をスタックに保存する処理の前の処理として、現在のＳＤＰをスタックに保存する処理を含めることも可能である。

【０２３２】

また図３のステップＳ２で新しいＤＰ（０ｘ０８８０）を設定する処理の前の処理として、新しいＳＤＰ（０ｘ０Ｆ００）を設定する処理を含めることも可能である。

【０２３３】

このように現在のＳＤＰをスタックに保存する処理を、引数Ａや引数Ｂをスタックに保存する処理よりも先に行うことによって、呼び出されるライブラリプログラムモジュールの側でこのスタックに保存されたＳＤＰの有無に関与しなくてもよいことは、前記第１の実施の形態と同じであるから詳細な説明を省略する。

【０２３４】

このような実施の形態によって、関数ｂａｒが、複数のアプリケーションプログラムモジュール（例えば、図４（ａ）のアプリケーションプログラムモジュール１２１とアプリケーションプログラムモジュール１３１）から同時に呼び出され、その結果再入が起こった時、明示的に、その外部変数の幾つかを共通に使用し、データの共有、受け渡し、その他の処理を行うことができる。

【０２３５】

またこの時、前記したセマフォ等を使った調停が行われることがある、或いは行わなければならないことも発生し得るが、本発明の内容と直接関係しないので説明を省略する。

【０２３６】

このように、ライブラリプログラムモジュールが複数のアプリケーションプログラムモジュールから呼び出されたとしても、各アプリケーションプログラムモジュール毎に設けられた実行固有データメモリ領域にアクセスするため、例えば外部変数へのアクセスで衝突が発生することは起こらず、プログラムモジュールを再入可能に構成することができる。また、ライブラリプログラムモジュールが複数のアプリケーションプログラムモジュールから呼び出されたとしても、各アプリケーションプログラムモジュール毎で共有する実行共有データメモリ領域にアクセスするため、例えば外部変数の幾つかを共通に利用することができる。

【０２３７】

また、ライブラリプログラムモジュール用の実行共有データメモリ領域、及びライブラリプログラムモジュール用の実行固有データメモリ領域のうちのいずれを使用するのかを特定するための記述が、ライブラリプログラムモジュールに含まれる外部変数毎に記述されるコモン宣言であるので、このコモン宣言の有無を判断することによって容易に実行共有データメモリ領域と実行固有データメモリ領域とを使い分けることができる。

【０２３８】

（第４の実施の形態）

図１６は、本発明の第４の実施の形態であるコンパイラが変換の対象とする呼び出し元プログラムであるアプリケーションプログラムモジュール１２１と、呼び出されるプログラムであるライブラリプログラムモジュール１５１と、その呼び出し処理時に行われる処理の一例を概念的に示す図である。図１６（ａ）は、各プログラムモジュールの固有のデータ領域へのアドレスを保存しているテーブルの一例を示す図であり、図１６（ｂ）は、

第４の実施の形態におけるアプリケーションプログラムモジュールのコード領域及びデータ領域の一例を示す図であり、図１６（ｃ）は、第４の実施の形態におけるライブラリプログラムモジュールのコード領域及びデータ領域の一例を示す図である。

【０２３９】

図１６（ａ）に示すように、第４の実施の形態では、呼び出し元アプリケーションプログラムモジュール単位、或いは呼び出し元アプリケーションプログラム単位、或いは呼び出し元アプリケーションプロセス単位、或いは呼び出し元アプリケーションタスク単位、或いは呼び出し元アプリケーション単位、或いは呼び出し元のプロセス単位に、或いはまた呼び出し元のタスク単位に、ＣＣＴ（Context Control Table）と呼ばれる、各プログラムモジュールの固有のデータ領域へのアドレスを保存しているテーブルを備えている。

【０２４０】

呼び出し元のアプリケーションプログラムモジュール１２１では、アプリケーションプログラム１２１用ＣＣＴを参照してアプリケーションプログラムモジュール１２１自体のデータ領域アドレスである「０ｘ６００」を取り出し、この値をＤＰに設定し、この値をオフセットアドレスとして使用して外部変数アクセスすることによって、アプリケーションプログラムモジュール１２１用の固有データ領域１２５を使用する。

【０２４１】

このアプリケーションプログラムモジュール１２１からライブラリプログラムモジュール１５１に含まれる関数ｂａｒを呼び出すには、まず関数ｂａｒに引き渡す引数Ａと引数Ｂをスタックに保存する。

【０２４２】

次に、アプリケーションプログラムモジュール１２１ではこの関数ｂａｒから自分自身へ帰ってくるためのアドレス、即ち関数ｂａｒへのジャンプ命令の次のアドレスをスタックに保存し、関数ｂａｒに制御を移行する。

【０２４３】

制御の移行を受けた呼び出し先ライブラリプログラムモジュール１５１に含まれる関数ｂａｒでは、現行のＤＰの内容を保存する。現行のＤＰの内容とは呼び出し元のアプリケーションプログラムモジュール１２１用の固有データ領域アドレスの値であり、関数ｂａｒから呼び出し元のプログラムに戻った時、再度この値をＤＰに設定してアプリケーションプログラムモジュール１２１用の固有データ領域を使用するために保存しておくものである。

【０２４４】

次にこのＤＰの内容を参照してアプリケーションプログラムモジュール１２１用ＣＣＴへのポインタアドレスを取り出し、更にそのポインタアドレスを参照してアプリケーションプログラムモジュール１２１用ＣＣＴを取り出し、更にそのＣＣＴの内容を参照して、関数ｂａｒでアプリケーションプログラムモジュール１２１が固有に使用するアプリケーションプログラムモジュール１２１用データ領域１５５のアドレス（この例では、０ｘ８００）を取り出し、その内容をＤＰレジスタに設定する。

【０２４５】

アプリケーションプログラムモジュール１２１によって呼び出されたライブラリプログラムモジュール１５１に含まれる関数ｂａｒでは、外部変数に対してアクセスする時、ＤＰレジスタの値を相対アドレスのベースアドレスとして使用してアクセスするので、関数ｂａｒが使用する全ての外部変数は０ｘ８００が相対アドレスのベースアドレスとして使用されてアクセスされ、アプリケーションプログラムモジュール１２１から呼び出される限りアプリケーションプログラムモジュール１２１用データ領域１５５が割り当てられる。

【０２４６】

これによって、複数のアプリケーションプログラムモジュールから関数ｂａｒに対して再入が発生したとしても、外部変数で衝突が発生することは起こらない。

【０２４７】

関数ｂａｒから呼び出し元のアプリケーションプログラムモジュール１２１に戻る時には、ＤＰの内容を保存しておいた呼び出し元のアプリケーションプログラムモジュール１２１用の固有データ領域アドレスの値に戻してから、リターンする。

【０２４８】

これによって、呼び出し元のアプリケーションプログラムモジュール１２１では再度、呼び出し元のアプリケーションプログラムモジュール１２１自体のデータ領域を使用することができる。

【０２４９】

第４の実施の形態では、ライブラリプログラムモジュール１５１の呼び出し側でＤＰの設定を行わないため、呼び出し側のオブジェクトプログラムのサイズを小さくすることができ、呼び出し回数が多い場合等に特に有効である。

【０２５０】

このように、アプリケーションプログラムモジュールからライブラリプログラムモジュールへの再入が発生したとしても、各プログラムモジュール用データメモリ領域の先頭アドレスを記憶するテーブル（ＣＣＴ）を介してアクセスするので、例えば外部変数へのアクセスで衝突が発生することは起こらず、プログラムモジュールを再入可能に構成することができる。

【０２５１】

また図示していないが、図１６（ｃ）で関数ｂａｒが「ｐｒｉｖａｔｅ」宣言が行われていれば、上記のようなＤＰの保存やアプリケーションプログラムモジュール１２１からの呼び出しで使用する関数ｂａｒ用のＤＰを、ＣＣＴから取り出して設定等の処理を行わないことは上記第２の実施の形態と同じであり、「ｐｕｂｌｉｃ」宣言が行われているならば明示的に、ＤＰの保存やアプリケーションプログラムモジュール１２１からの呼び出しで使用する関数ｂａｒ用のＤＰをＣＣＴから取り出して設定等の処理等が行われることを示す。これらは前記第２の実施の形態と同じであるから説明は省略する。

【０２５２】

この場合、アプリケーションプログラムモジュールに含まれるプログラムコード領域から呼び出されるライブラリプログラムモジュールに含まれるプログラムコード領域の記述に基づいて、処理を短縮するか否かが判別される。したがって、必ずしもアプリケーションプログラムモジュールと呼び出されるライブラリプログラムモジュールとが異なるデータ領域を使用することが必要でないケースでは、各プログラムモジュール用データ領域の先頭アドレスを記憶したテーブル（ＣＣＴ）を介した先頭アドレスの設定の処理を省略し、オブジェクトプログラムの実行を高速化することができ、不要な処理を出力しないことでオブジェクトプログラムのサイズを小さくすることができる。

【０２５３】

更にまた図示していないが、このＣＣＴには前記ＤＰだけでなく、前記第３の実施の形態で説明した新しいＳＤＰ（０ｘ０Ｆ００）を、図７（ａ）の固有データ領域１２５に格納して呼び出しプログラムモジュール１２１のプログラムコード領域１２３でその保存や設定を行うのではなく、例えば、図１６（ａ）のＣＣＴと同様のテーブルを用意して、そのテーブルに各プログラムモジュールの共有データ領域の先頭アドレスを格納しておき、呼び出されるライブラリプログラムモジュールである、例えば図１６（ｃ）のライブラリプログラムモジュール１５１のプログラムコード領域１５３における処理の中で、そのＣＣＴと同様のテーブルと、ＣＣＴと同様のテーブルに格納されたＳＤＰとを読み出して設定することもできる。これについても前記ＤＰの保存と設定と同じであるから詳細な説明は省略する。また、ＣＣＴと同様のテーブルは、ＣＣＴと異なり、実行環境で１つだけ保持するとしてもよい。

【０２５４】

（第５の実施の形態）

次に、本発明の第５の実施の形態について説明する。本発明の第５の実施の形態では、プログラムモジュールに識別番号を付与し、当該識別番号と各プログラムモジュールのデ

ータ領域のＤＰとを対応付けたテーブルを作成し、アプリケーションプログラムモジュールからライブラリプログラムモジュールが呼び出されると、各プログラムモジュールの識別番号をインデックスとしてテーブルを参照して、ライブラリプログラムモジュールのデータ領域のＤＰと、アプリケーションプログラムモジュールのデータ領域のＤＰとを切り替える。

【０２５５】

図１７は、本発明の第５の実施の形態におけるコンパイル装置の構成を示すブロック図である。図１７に示すコンパイル装置１４は、プログラム実行部１６及びプログラム記憶部１７を備えて構成される。

【０２５６】

プログラム実行部１６は、例えばＣＰＵ（中央演算処理装置）等で構成され、アドレス保存プログラム生成部１６ａ、アドレス設定プログラム生成部１６ｂ、移行プログラム生成部１６ｃ、アドレス再設定プログラム生成部１６ｄ、アクセスプログラム生成部１６ｅ、識別番号付与プログラム生成部１６ｇ、テーブル作成プログラム生成部１６ｈ、識別番号取得プログラム生成部１６ｉ及び先頭アドレス取得プログラム生成部１６ｊを備えて構成される。プログラム記憶部１７は、コンパイラプログラム記憶部１７ａ、ソースプログラム記憶部１７ｂ及びオブジェクトプログラム記憶部１７ｃを備えて構成される。

【０２５７】

なお、図１７に示す第５の実施の形態におけるコンパイル装置と、図２に示す第１の実施の形態におけるコンパイル装置とで同じ構成物については、同じ符号を付し、以下の説明では第１の実施の形態とは異なる構成についてのみ説明する。

【０２５８】

識別番号付与プログラム生成部１６ｇは、呼び出し元プログラムモジュールを識別する識別番号と、他プログラムモジュールを識別する識別番号とをそれぞれに付与するための識別番号付与プログラムを生成する。

【０２５９】

テーブル作成プログラム生成部１６ｈは、付与された呼び出し元プログラムモジュールの識別番号と、呼び出し元プログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、付与された他プログラムモジュールの識別番号と、他プログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するためのテーブル作成プログラムを生成する。

【０２６０】

識別番号取得プログラム生成部１６ｉは、呼び出し元プログラムモジュールから他プログラムモジュールへの呼び出し命令を受けて、他プログラムモジュールの識別番号を取得するための識別番号取得プログラムを生成する。

【０２６１】

先頭アドレス取得プログラム生成部１６ｊは、取得された識別番号をインデクスとして、作成されたテーブルから他プログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムを生成する。

【０２６２】

図１８は、本発明の第５の実施の形態における端末装置の構成を示すブロック図である。図１８に示す端末装置１２は、プログラム実行部１８、メモリ１９、データ読込部２０及び通信部２１を備えて構成される。通信部２１は、サーバ１１から送信されるオブジェクトプログラムをダウンロードするものであり、識別番号付与部２１ａを備えて構成される。データ読込部２０は、通信部２１によってダウンロードされたオブジェクトプログラムをメモリ１９に読み込むものであり、ＤＰテーブル作成部２０ａを備えて構成される。プログラム実行部１８は、例えばＤＳＰ等で構成され、アドレス保存部１８ａ、アドレス設定部１８ｂ、移行部１８ｃ、アドレス再設定部１８ｄ、アクセス部１８ｅ、識別番号取得部１８ｇ及び先頭アドレス取得部１８ｈを備えて構成される。メモリ１９は、アプリケーションプログラムモジュール記憶部１９ａ、ライブラリプログラムモジュール記憶部１

９ｂ、スタック１９ｃ及びＤＰテーブル記憶部１９ｄを備えて構成される。

【０２６３】

なお、図１８に示す第５の実施の形態における端末装置と、図５に示す第１の実施の形態における端末装置とで同じ構成物については、同じ符号を付し、以下の説明では第１の実施の形態とは異なる構成についてのみ説明する。

【０２６４】

識別番号付与部２１ａは、呼び出し元プログラムモジュールであるアプリケーションプログラムモジュールを識別する識別番号を当該アプリケーションプログラムモジュールに付与するとともに、他プログラムモジュールであるライブラリプログラムモジュールを識別する識別番号を当該ライブラリプログラムモジュールに付与する。

【０２６５】

ＤＰテーブル作成部２０ａは、識別番号付与部２１ａによって付与されたアプリケーションプログラムモジュールの識別番号と、アプリケーションプログラムモジュールのデータ領域の先頭アドレス（ＤＰ）とを対応付けるとともに、識別番号付与部２１ａによって付与されたライブラリプログラムモジュールの識別番号と、ライブラリプログラムモジュールのデータ領域の先頭アドレスとを対応付けるＤＰテーブルを作成し、ＤＰテーブル記憶部１９ｄに記憶する。

【０２６６】

識別番号取得部１８ｇは、アプリケーションプログラムモジュールからライブラリプログラムモジュールへの呼び出し命令を受けて、ライブラリプログラムモジュールの識別番号を取得する。

【０２６７】

先頭アドレス取得部１８ｈは、識別番号取得部１８ｇによって取得された識別番号をインデクスとして、ＤＰテーブル作成部２０ａによって作成されたＤＰテーブルからライブラリプログラムモジュールのデータ領域の先頭アドレスを取得する。

【０２６８】

アドレス設定部１８ｂは、アプリケーションプログラムモジュールによって呼び出されるライブラリプログラムモジュールが使用するライブラリプログラムモジュール用データ領域の先頭アドレスを、先頭アドレス取得部１８ｈによって取得された先頭アドレスに設定する。

【０２６９】

ＤＰテーブル記憶部１９ｄは、ＤＰテーブル作成部２０ａによって作成されたＤＰテーブルを記憶する。

【０２７０】

図１９は、図１８に示す端末装置による動的モジュールリンク処理の一例を示すフローチャートである。

【０２７１】

ステップＳ３１において、通信部２１は、サーバ１１のコンパイル装置１４によってソースプログラムからオブジェクトプログラムに変換されたアプリケーションプログラムモジュール及びライブラリプログラムモジュールをダウンロードする。

【０２７２】

ステップＳ３２において、識別番号付与部２１ａは、ダウンロードされたアプリケーションプログラムモジュール及びライブラリプログラムモジュールのそれぞれに対して固有の識別番号を付与する。識別番号付与部２１ａによって付与された識別番号は、アプリケーションプログラムモジュール記憶部１９ａ及びライブラリプログラムモジュール記憶部１９ｂにそれぞれ記憶される。

【０２７３】

ステップＳ３３において、データ読込部２０は、アプリケーションプログラムモジュール及びライブラリプログラムモジュールのコード領域を読み込む。

【０２７４】

ステップＳ３４において、ＤＰテーブル作成部２０ａは、メモリ１９のＤＰテーブル記憶部１９ｄに、各プログラムモジュールのデータ領域の先頭アドレスと、各プログラムモジュールの識別番号とが対応付けられたＤＰテーブルを作成する。なお、この時点ではＤＰテーブルにはデータがまだ入力されておらず、メモリ内の領域のみが確保される。

【０２７５】

ステップＳ３５において、データ読込部２０は、アプリケーションプログラムモジュール及びライブラリプログラムモジュールにデータ領域を割り当てる。すなわち、アプリケーションプログラムモジュール及びライブラリプログラムモジュールのデータ領域を、メモリ１９のアプリケーションプログラムモジュール記憶部１９ａ及びライブラリプログラムモジュール記憶部１９ｂにそれぞれ記憶する。

【０２７６】

ステップＳ３６において、ＤＰテーブル作成部２０ａは、アプリケーションプログラムモジュールのデータ領域の先頭アドレス（ＤＰ）をＤＰテーブル記憶部１９ｄに記憶されているアプリケーションプログラムモジュールの識別番号に対応付けて設定し、ライブラリプログラムモジュールのデータ領域の先頭アドレスをＤＰテーブル記憶部１９ｄに記憶されているライブラリプログラムモジュールの識別番号に対応付けて設定する。

【０２７７】

ステップＳ３７において、識別番号取得部１８ｇは、アプリケーションプログラムモジュールからライブラリプログラムモジュールが呼び出されたか否かを判断する。ここで、アプリケーションプログラムモジュールからライブラリプログラムモジュールが呼び出されたと判断されると、ステップＳ３８に移行し、アプリケーションプログラムモジュールからライブラリプログラムモジュールが呼び出されていないと判断されると、アプリケーションプログラムモジュールからライブラリプログラムモジュールが呼び出されるまで待機状態となる。

【０２７８】

ステップＳ３８において、アドレス保存部１８ａは、アプリケーションプログラムモジュールのコード領域の先頭アドレス及びデータ領域の先頭アドレスをメモリのスタック１９ｃに保存することによって退避させる。

【０２７９】

ステップＳ３９において、識別番号取得部１８ｇは、ライブラリプログラムモジュール記憶部１９ｂに記憶されているライブラリプログラムモジュールの識別番号を取得する。

【０２８０】

ステップＳ４０において、先頭アドレス取得部１８ｈは、識別番号取得部１８ｇによって取得されたライブラリプログラムモジュールの識別番号をインデクスとしてＤＰテーブル記憶部１９ｄに記憶されているＤＰテーブルを参照し、当該ライブラリプログラムモジュールの識別番号に対応する先頭アドレスを取得する。

【０２８１】

ステップＳ４１において、アドレス設定部は、アプリケーションプログラムモジュールのデータ領域の先頭アドレスを、先頭アドレス取得部１８ｈによって取得されたライブラリプログラムモジュールのデータ領域の先頭アドレスに設定することによって切り替える。

【０２８２】

ステップＳ４２において、アドレス再設定部１８ｄは、アドレス保存部によってメモリ１９のスタック１９ｃに退避させたアプリケーションプログラムモジュールのコード領域の先頭アドレス及びデータ領域の先頭アドレスを再設定することによって復元する。

【０２８３】

このように、まず、識別番号付与プログラム生成部１６ｇによって、アプリケーションプログラムモジュールを識別する識別番号と、ライブラリプログラムモジュールを識別する識別番号とをそれぞれに付与するための識別番号付与プログラムが生成される。そして、テーブル作成プログラム生成部１６ｈによって、付与されたアプリケーションプログラ

ムモジュールの識別番号と、アプリケーションプログラムモジュールのデータ領域の先頭アドレスとを対応付けるとともに、付与されたライブラリプログラムモジュールの識別番号と、ライブラリプログラムモジュールのデータ領域の先頭アドレスとを対応付けるテーブルを作成するためのテーブル作成プログラムが生成される。アプリケーションプログラムモジュールからライブラリプログラムモジュールへの呼び出し命令を受けて、識別番号取得プログラム生成部１６ｉによって、ライブラリプログラムモジュールの識別番号を取得するための識別番号取得プログラムが生成される。先頭アドレス取得プログラム生成部１６ｊによって、取得された識別番号をインデクスとして、作成されたテーブルからライブラリプログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムが生成される。さらに、アドレス設定プログラム生成部１６ｂによって、取得されたライブラリプログラムモジュールのデータ領域の先頭アドレスと、アプリケーションプログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムが生成される。

【０２８４】

したがって、従来のＭＭＵ（Memory Management Unit）のようにアドレス変換するためのテーブルをページ単位で作成するのではなく、モジュール単位で作成することによって、テーブルのデータ量を削減することができ、資源制約の厳しい小型の機器に適合させることができる。また、テーブルのデータ量を削減することができるので、従来のＭＭＵによるアドレス変換に比して実行速度を向上させることができる。

【０２８５】

なお、コンパイル装置１４のプログラム実行部１６は、ライブラリプログラムモジュール外で定義された変数又は関数へのアドレスを保持するためのアドレス保持プログラムを生成するアドレス保持プログラム生成部を備え、端末装置１２のプログラム実行部１８は、ライブラリプログラムモジュール外で定義された変数又は関数へのアドレスを保持するアドレス保持部を備えてもよい。この場合、アクセス部１８ｅは、ライブラリプログラムモジュール内で定義された変数へアクセスする場合、先頭アドレス取得部１８ｈによって取得された先頭アドレスからの相対アドレスでアクセスし、ライブラリプログラムモジュール内で定義された関数へアクセスする場合、ＰＣ（プログラムカウンタ）からの相対アドレスでアクセスする。また、アクセス部１８ｅは、ライブラリプログラムモジュール外で定義された変数又は関数へアクセスする場合、アドレス保持部によって保持されたアドレスに、先頭アドレス取得部１８ｈによって取得された先頭アドレスからの相対アドレスでアクセスした後、アドレス保持部によって保持されたアドレス経由で間接アクセスする。

【０２８６】

この構成によれば、アドレス保持プログラム生成部によって、ライブラリプログラムモジュール外で定義された変数又は関数へのアドレスを保持するためのアドレス保持プログラムが生成される。そして、アクセスプログラム生成部１６ｅによって、ライブラリプログラムモジュール内で定義された変数又は関数へアクセスする場合、取得された先頭アドレスからの相対アドレスでアクセスされ、ライブラリプログラムモジュール内で定義された関数へアクセスする場合、プログラムカウンタからの相対アドレスでアクセスされ、ライブラリプログラムモジュール外で定義された変数又は関数へアクセスする場合、保持されたアドレスに取得された先頭アドレスからの相対アドレスでアクセスした後、保持されたアドレス経由で間接アクセスするためのアクセスプログラムが生成される。したがって、ライブラリプログラムモジュール外で定義された変数又は関数へアクセスする場合に、保持されたアドレス経由で間接的にアクセスすることができる。

【０２８７】

なお、本実施の形態では、コンパイル装置１４が備えるテーブル作成プログラム生成部１６ｈによって生成されたテーブル作成プログラムを端末装置１２がダウンロードすることによって、データ読込部２０がＤＰテーブル作成部２０ａとして機能し、識別番号付与プログラム生成部１６ｇによって生成された識別番号付与プログラムを端末装置１２がダ

ウンロードすることによって、通信部２１が識別番号付与部２１ａとして機能している。しかしながら、あらかじめ、データ読込部２０がＤＰテーブル作成部２０ａとしての機能を備えていてもよく、通信部２１が識別番号付与部２１ａとしての機能を備えていてもよい。この場合、図１７に示すコンパイル装置１４のプログラム実行部１６は、アドレス保存プログラム生成部１６ａ、アドレス設定プログラム生成部１６ｂ、移行プログラム生成部１６ｃ、アドレス再設定プログラム生成部１６ｄ、アクセスプログラム生成部１６ｅ、識別番号取得プログラム生成部１６ｉ及び先頭アドレス取得プログラム生成部１６ｊのみを備えて構成される。

【０２８８】

このように、識別番号取得プログラム生成部１６ｉによって、アプリケーションプログラムモジュールからライブラリプログラムモジュールへの呼び出し命令を受けて、ライブラリプログラムモジュールを識別する識別番号を取得するための識別番号取得プログラムが生成される。先頭アドレス取得プログラム生成部１６ｊによって、取得された識別番号をインデクスとして、アプリケーションプログラムモジュールの識別番号と、アプリケーションプログラムモジュールのデータ領域の先頭アドレスとが対応付けられるとともに、ライブラリプログラムモジュールの識別番号と、ライブラリプログラムモジュールのデータ領域の先頭アドレスとが対応付けられたテーブルから、ライブラリプログラムモジュールのデータ領域の先頭アドレスを取得するための先頭アドレス取得プログラムが生成される。そして、アドレス再設定プログラム生成部１６ｄによって、取得されたライブラリプログラムモジュールのデータ領域の先頭アドレスと、アプリケーションプログラムモジュールのデータ領域の先頭アドレスとを切り替えるための先頭アドレス切替プログラムが生成される。

【０２８９】

したがって、アドレス変換するためのテーブルを、ページ単位ではなくモジュール単位で行うことによって、テーブルのデータ量を削減することができ、資源制約の厳しい小型の機器に適合させることができる。また、識別番号を付与するプログラムやテーブルを作成するプログラムをコンパイルする必要がないため、コンパイル処理に要する時間を短縮することができ、さらに、端末装置１２に送信するプログラムモジュールのデータ量を削減することができる。

【０２９０】

次に、メモリのフラグメンテーション（細分化）の解消方法について説明する。メモリに対して書き込みや削除を繰り返し行うと、メモリ内の空き領域が細分化されてしまい、１つのデータを連続してメモリに記憶させることが困難となる。この場合、データを複数の領域に分割して記憶させなければならず、メモリへのアクセス速度の低下に繋がる。そのため、細分化されたメモリ領域をひとまとめにし、連続したメモリ領域にする必要がある。

【０２９１】

図２０は、フラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。なお、他の実施の形態と同じ構成物については、同じ符号を付し、説明を省略する。

【０２９２】

図２０に示すように、サーバ１１は、通信部２２及びコンパイル装置１４を備えて構成される。通信部２２は、コンパイル装置１４によって変換されたオブジェクトプログラムを端末装置１２に送信する。コンパイル装置１４は、プログラム実行部１６及びプログラム記憶部１７を備えて構成される。プログラム実行部１６は、アプリケーション終了プログラム生成部２３ａ、コンパクションプログラム生成部２３ｂ及びアプリケーション再開プログラム生成部２３ｃを備えて構成される。

【０２９３】

アプリケーション終了プログラム生成部２３ａは、少なくとも１つのプログラムモジュールで構成されるとともに、現在実行中であるアプリケーションを終了するための終了プ

ログラムを生成する。

【０２９４】

コンパクションプログラム生成部２３ｂは、アプリケーションが終了された後、メモリ１９をコンパクションするためのコンパクションプログラムを生成する。なお、コンパクションとは、細かく分断されたメモリ１９内の領域をひとまとめにし、それらを連続した領域にすることである。

【０２９５】

アプリケーション再開プログラム生成部２３ｃは、メモリ１９がコンパクションされた後、終了されたアプリケーションを最初から再開するための再開プログラムを生成する。

【０２９６】

端末装置１２は、プログラム実行部１８、メモリ１９及び通信部２１を備えて構成される。通信部２１は、サーバ１１より送信されたオブジェクトプログラムを受信し、メモリ１９に記憶する。プログラム実行部１８は、アプリケーション終了部２４ａ、コンパクション部２４ｂ及びアプリケーション再開部２４ｃを備えて構成される。

【０２９７】

アプリケーション終了部２４ａは、少なくとも１つのプログラムモジュールで構成されるとともに、現在実行中であるアプリケーションを終了する。コンパクション部２４ｂは、アプリケーション終了部２４ａによってアプリケーションが終了された後、メモリ１９をコンパクションする。アプリケーション再開部２４ｃは、メモリ１９がコンパクションされた後、アプリケーション終了部２４ａによって終了されたアプリケーションを最初から再開する。

【０２９８】

このように、アプリケーション終了プログラム生成部２３ａによって、少なくとも１つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するための終了プログラムが生成される。そして、アプリケーションが終了された後、コンパクションプログラム生成部２３ｂによって、メモリ１９をコンパクションするためのコンパクションプログラムが生成され、メモリ１９がコンパクションされた後、アプリケーション再開プログラム生成部２３ｃによって、アプリケーションを最初から再開するための再開プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【０２９９】

次に、メモリのフラグメンテーションを解消する第１の変形例について説明する。第１の変形例では、アドレス値に依存しないような情報で現在のアプリケーションの状態を保存し、コンパクション後、その状態まで実行させるものである。

【０３００】

図２１は、第１の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。なお、他の実施の形態と同じ構成物については、同じ符号を付し、説明を省略する。

【０３０１】

図２１に示すように、サーバ１１は、通信部２２及びコンパイル装置１４を備えて構成される。コンパイル装置１４は、プログラム実行部１６及びプログラム記憶部１７を備えて構成される。プログラム実行部１６は、アプリケーション終了プログラム生成部２３ａ、アプリケーション情報保存プログラム生成部２３ｄ、コンパクションプログラム生成部２３ｂ及びアプリケーション実行プログラム生成部２３ｅを備えて構成される。

【０３０２】

アプリケーション情報保存プログラム生成部２３ｄは、終了されるまでのアプリケーションのアドレス値に依存しない情報を保存するための保存プログラムを生成する。

【０３０３】

アプリケーション実行プログラム生成部２３ｅは、メモリ１９がコンパクションされた後、保存された情報を読み出し、読み出された情報に基づいてアプリケーションが終了し

た時点における状態まで当該アプリケーションを実行するための実行プログラムを生成する。

【０３０４】

端末装置１２は、プログラム実行部１８、メモリ１９及び通信部２１を備えて構成される。プログラム実行部１８は、アプリケーション終了部２４ａ、アプリケーション情報保存部２４ｄ、コンパクション部２４ｂ及びアプリケーション実行部２４ｅを備えて構成される。

【０３０５】

アプリケーション情報保存部２４ｄは、アプリケーション終了部２４ａによって終了されるまでのアプリケーションのアドレス値に依存しない情報を保存する。アプリケーション実行部２４ｅは、コンパクション部２４ｂによってメモリ１９がコンパクションされた後、アプリケーション情報保存部２４ｄによって保存された情報を読み出し、読み出された情報に基づいてアプリケーションが終了した時点における状態まで当該アプリケーションを実行する。

【０３０６】

このように、アプリケーション終了プログラム生成部２３ａによって、少なくとも１つのモジュールで構成され、かつ現在実行中であるアプリケーションを終了するためのアプリケーション終了プログラムが生成される。そして、アプリケーション情報保存プログラム生成部２３ｄによって、終了されるまでのアプリケーションのアドレス値に依存しない情報を保存するためのアプリケーション情報保存プログラムが生成される。コンパクションプログラム生成部２３ｂによって、メモリ１９をコンパクションするためのコンパクションプログラムが生成される。メモリ１９がコンパクションされた後、アプリケーション実行プログラム生成部２３ｅによって、保存された情報を読み出し、読み出された情報に基づいてアプリケーションが終了した時点における状態まで当該アプリケーションを実行するためのアプリケーション実行プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【０３０７】

次に、メモリのフラグメンテーションを解消する第２の変形例について説明する。第２の変形例では、全メモリ空間に対してリファレンスフラグ（識別子）の領域を設け、メモリアクセスの際に、アドレス値を設定する場合と、数値等のアドレス値以外を設定する場合とをリファレンスフラグによって識別し、コンパクション後、リファレンスフラグからアドレス値が設定されているエントリについてリロケーション（再配置）を行うものである。

【０３０８】

図２２は、第２の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。なお、他の実施の形態と同じ構成物については、同じ符号を付し、説明を省略する。

【０３０９】

図２２に示すように、サーバ１１は、通信部２２及びコンパイル装置１４を備えて構成される。コンパイル装置１４は、プログラム実行部１６及びプログラム記憶部１７を備えて構成される。プログラム実行部１６は、メモリ状態記憶プログラム生成部２３ｆ、コンパクションプログラム生成部２３ｂ及び再配置プログラム生成部２３ｇを備えて構成される。

【０３１０】

メモリ状態記憶プログラム生成部２３ｆは、コンパクション前のメモリ１９の状態（メモリマップ）を記憶するためのメモリ状態記憶プログラムを生成する。

【０３１１】

再配置プログラム生成部２３ｇは、メモリ１９がコンパクションされた後、アドレス値を設定する場合と、数値等のアドレス値以外を設定する場合とを識別するリファレンスフラグに基づいてアドレス値が設定されているエントリに対して、記憶されているコンパク

ション前のメモリ１９の状態を参照して、アドレス値の再配置を行うための再配置プログラムを生成する。

【０３１２】

端末装置１２は、プログラム実行部１８、メモリ１９及び通信部２１を備えて構成される。プログラム実行部１８は、メモリ状態記憶部２４ｆ、コンパクション部２４ｂ及び再配置部２４ｇを備えて構成される。

【０３１３】

メモリ状態記憶部２４ｆは、コンパクション前のメモリ１９の状態（メモリマップ）を記憶する。再配置部２４ｇは、メモリ１９がコンパクションされた後、アドレス値を設定する場合と、数値等のアドレス値以外を設定する場合とを識別するリファレンスフラグに基づいてアドレス値が設定されているエントリに対して、記憶されているコンパクション前のメモリ１９の状態を参照して、アドレス値の再配置を行う。

【０３１４】

図２３は、第２の変形例におけるフラグメンテーションの解消方法について説明するための図であり、図２３（ａ）は、領域が細分化されたコンパクション前のメモリの状態を示す図であり、図２３（ｂ）は、コンパクション後のメモリの状態を示す図であり、図２３（ｃ）は、リロケーション後のメモリの状態を示す図である。

【０３１５】

図２３（ａ）に示すメモリ１９は、アドレスが０ｘ３００から０ｘ４００までの領域が未使用となっており、０ｘ５５０にアドレス値である０ｘ５５８が格納されており、０ｘ５５４に３という数値が格納されており、０ｘ５５８に１００という数値が格納されており、０ｘ７００から未使用の領域となっていおり、その他の領域には他のデータが格納されている。本変形例では、メモリ１９の全メモリ空間に対してリファレンスフラグの領域ＲＦが設けられている。この領域ＲＦには、アドレス値が設定される場合は「１」が設定され、数値等のアドレス値以外が設定される場合は「０」が設定される。図２３（ａ）では、アドレス０ｘ５５０には、ポインタ変数としてアドレス値０ｘ５５８が設定されているので、リファレンスフラグが１に設定されており、アドレス０ｘ５５４及び０ｘ５５８には、数値３及び１００が設定されているので、リファレンスフラグが０に設定されている。

【０３１６】

図２３（ａ）に示す状態のメモリ１９をコンパクションすると、図２３（ｂ）に示す状態となる。すなわち、図２３（ｂ）に示すように、メモリ１９は、未使用の領域であった０ｘ３００から０ｘ４００までの領域がなくなり、０ｘ１００から０ｘ６００までの領域がひとまとまりとなり、０ｘ６００から未使用の領域となっている。この場合、メモリ１９内のアドレスがずれるため、コンパクション前にアドレス値０ｘ５５８を格納していたアドレス０ｘ５５０は０ｘ４５０に移動し、アドレス０ｘ５５０がポイントしていたアドレス０ｘ５５８も０ｘ４５８に移動している。この結果、図２３（ｂ）に示すように、アドレス０ｘ４５０に格納されているポインタ変数がポイントするアドレスがなくなってしまう。

【０３１７】

そこで、メモリ状態記憶部２４ｆは、コンパクション前のメモリ１９の状態を記憶しておき、再配置部２４ｇは、メモリ１９がコンパクションされた後、リファレンスフラグの値を識別し、当該リファレンスフラグが１に設定されているエントリに対して、記憶されているコンパクション前のメモリの状態とコンパクション後のメモリ１９の状態とを参照して、アドレス値の再配置を行う。例えば、図２３（ｃ）に示すように、アドレス０ｘ４５０に格納されているアドレス値は、０ｘ５５８から正しいアドレス値である０ｘ４５８に再配置される。

【０３１８】

このように、メモリ１９内において、アドレス値を設定するか否かを示す識別子を設け、メモリ状態記憶プログラム生成部２３ｆによって、コンパクション前のメモリ１９の状

態を記憶するためのメモリ状態記憶プログラムが生成される。そして、コンパクションプログラム生成部２３ｂによって、メモリ１９をコンパクションするためのコンパクションプログラムが生成される。さらに、メモリ１９がコンパクションされた後、再配置プログラム生成部２３ｇによって、識別子に基づいてアドレス値が設定されているエントリに対して、記憶されているコンパクション前のメモリ１９の状態を参照して、アドレス値の再配置を行うための再配置プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【０３１９】

次に、メモリのフラグメンテーションを解消する第３の変形例について説明する。第３の変形例では、メモリ内の領域を、アドレス値が格納されるアドレス値用メモリ領域と、数値等のアドレス値以外が格納される非アドレス値用メモリ領域とに分割し、コンパクション後、アドレス値用メモリ領域のみについてアドレス値のリロケーション（再配置）を行うものである。

【０３２０】

図２４は、第３の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。なお、他の実施の形態と同じ構成物については、同じ符号を付し、説明を省略する。

【０３２１】

図２４に示すように、サーバ１１は、通信部２２及びコンパイル装置１４を備えて構成される。コンパイル装置１４は、プログラム実行部１６及びプログラム記憶部１７を備えて構成される。プログラム実行部１６は、メモリ状態記憶プログラム生成部２３ｈ、コンパクションプログラム生成部２３ｂ及びアドレス値用メモリ再配置プログラム生成部２３ｉを備えて構成される。

【０３２２】

メモリ状態記憶プログラム生成部２３ｈは、アドレス値を格納するアドレス値用メモリ領域及びアドレス値以外を格納する非アドレス値用メモリ領域のコンパクション前の状態を記憶するためのメモリ状態記憶プログラムを生成する。

【０３２３】

アドレス値用メモリ再配置プログラム生成部２３ｉは、アドレス値用メモリ領域及び非アドレス値用メモリ領域がコンパクションされた後、記憶されているコンパクション前のアドレス値用メモリ領域の状態を参照して、アドレス値用メモリ領域のみのアドレス値の再配置を行うための再配置プログラムを生成する。

【０３２４】

端末装置１２は、プログラム実行部１８、メモリ１９及び通信部２１を備えて構成される。プログラム実行部１８は、メモリ状態記憶部２４ｈ、コンパクション部２４ｂ及びアドレス値用メモリ再配置部２４ｉを備えて構成される。

【０３２５】

メモリ１９は、アドレス値を格納するアドレス値用メモリ領域と、数値等のアドレス値以外を格納する非アドレス値用メモリ領域に分割されている。

【０３２６】

メモリ状態記憶部２４ｈは、アドレス値を格納するアドレス値用メモリ領域及びアドレス値以外を格納する非アドレス値用メモリ領域のコンパクション前の状態を記憶する。

【０３２７】

アドレス値用メモリ再配置部２４ｉは、アドレス値用メモリ領域及びアドレス値以外を格納する非アドレス値用メモリ領域がコンパクションされた後、メモリ状態記憶部２４ｈによって記憶されているコンパクション前のアドレス値用メモリ領域の状態を参照して、アドレス値用メモリ領域のみのアドレス値の再配置を行う。

【０３２８】

図２５は、第３の変形例におけるフラグメンテーションの解消方法について説明するための図であり、図２５（ａ）は、領域が細分化されたコンパクション前のメモリの状態を

示す図であり、図２５（ｂ）は、コンパクション後のメモリの状態を示す図であり、図２５（ｃ）は、リロケーション後のメモリの状態を示す図である。

【０３２９】

図２５（ａ）に示すように、メモリ１９は、アドレス値を格納しているアドレス値用メモリ領域と、数値等のアドレス値以外を格納している非アドレス値用メモリ領域とに分割されている。

【０３３０】

図２５（ａ）に示すメモリ１９のアドレス値用メモリ領域は、アドレスが０ｘ１００から０ｘ２００までの領域が未使用となっており、０ｘ３００にアドレス値である０ｘ３０４が格納されており、０ｘ３０４に０ｘ２００が格納されており、０ｘ５００から未使用の領域となっており、その他の領域には他のデータが格納されている。また、メモリ１９の非アドレス値用メモリ領域は、アドレスが０ｘ５０００から０ｘ５１００までの領域が未使用となっており、０ｘ５２００に１０００という数値が格納されており、０ｘ５２０４に２０００という数値が格納されており、０ｘ５４００から未使用の領域となっていおり、その他の領域には他のデータが格納されている。

【０３３１】

図２５（ａ）に示す状態のメモリ１９をコンパクションすると、図２５（ｂ）に示す状態となる。すなわち、図２５（ｂ）に示すように、メモリ１９のアドレス値用メモリ領域は、未使用の領域であった０ｘ１００から０ｘ２００までの領域がなくなり、０ｘ１００から０ｘ４００までの領域がひとまとまりとなり、０ｘ４００から未使用の領域となっている。また、メモリ１９の非アドレス値用メモリ領域は、未使用の領域であった０ｘ５０００から０ｘ５１００までの領域がなくなり、０ｘ５０００から０ｘ５３００までの領域がひとまとまりとなり、０ｘ５３００から未使用の領域となっている。

【０３３２】

この場合、メモリ１９内のアドレスがずれるため、コンパクション前にアドレス値０ｘ３０４を格納していたアドレス０ｘ３００は０ｘ２００に移動し、コンパクション前にアドレス値０ｘ２００を格納していたアドレス０ｘ３０４は０ｘ２０４に移動し、アドレス０ｘ３０４がポイントしていたアドレス０ｘ２００も０ｘ１００に移動している。この結果、図２５（ｂ）に示すように、アドレス０ｘ２００及び０ｘ２０４に格納されているポインタ変数がポイントするアドレスがなくなってしまう。

【０３３３】

そこで、メモリ状態記憶部２４ｈは、メモリ１９のアドレス値用メモリ領域及び非アドレス値用メモリ領域のコンパクション前の状態を記憶しておき、アドレス値用メモリ再配置部２４ｉは、メモリ１９がコンパクションされた後、メモリ状態記憶部２４ｈによって記憶されているコンパクション前のメモリ１９のアドレス値用メモリ領域の状態とコンパクション後のメモリ１９のアドレス値用メモリ領域の状態とを参照して、アドレス値用メモリ領域のみのアドレス値の再配置を行う。例えば、図２５（ｃ）に示すように、アドレス値用メモリ再配置部２４ｉによって、アドレス０ｘ２００に格納されているアドレス値は、０ｘ３０４から正しいアドレス値である０ｘ２０４に再配置され、アドレス０ｘ２０４に格納されているアドレス値は、０ｘ２００から正しいアドレス値である０ｘ１００に再配置される。

【０３３４】

このように、アドレス値を格納するアドレス値用メモリ領域と、アドレス値以外を格納する非アドレス値用メモリ領域とを設け、メモリ状態記憶プログラム生成部２３ｈによって、コンパクション前のアドレス値用メモリ領域及び非アドレス値用メモリ領域の状態を記憶するためのメモリ状態記憶プログラムが生成される。コンパクションプログラム生成部２３ｂによって、アドレス値用メモリ領域及び非アドレス値用メモリ領域をコンパクションするためのコンパクションプログラムが生成される。アドレス値用メモリ領域及び非アドレス値用メモリ領域がコンパクションされた後、アドレス値用メモリ再配置プログラム生成部２３ｉによって、記憶されているコンパクション前のアドレス値用メモリ領域の

状態を参照して、アドレス値用メモリ領域のみのアドレス値の再配置を行うためのアドレス値用メモリ再配置プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【０３３５】

次に、メモリのフラグメンテーションを解消する第４の変形例について説明する。第４の変形例では、メモリ内の領域を、アドレス値が格納されるアドレス値用メモリ領域と、数値等のアドレス値以外が格納される非アドレス値用メモリ領域とに分割し、メモリ内のポインタ変数に対して、モジュールを識別するための識別番号と、モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグ（識別子）とを設け、モジュールの実行時において、ポインタ変数にアドレスを代入する際に、識別番号と識別フラグとを設定し、コンパクション後、アドレス値用メモリ領域のみについて、識別番号及び識別フラグに基づいてアドレス値を再計算するものである。

【０３３６】

図２６は、第４の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。なお、他の実施の形態と同じ構成物については、同じ符号を付し、説明を省略する。

【０３３７】

図２６に示すように、サーバ１１は、通信部２２及びコンパイル装置１４を備えて構成される。コンパイル装置１４は、プログラム実行部１６及びプログラム記憶部１７を備えて構成される。プログラム実行部１６は、識別番号及び識別フラグ設定プログラム生成部２３ｊ、コンパクションプログラム生成部２３ｂ及び再計算プログラム生成部２３ｋを備えて構成される。

【０３３８】

識別番号及び識別フラグ設定プログラム生成部２３ｊは、プログラムモジュールの実行時において、ポインタにアドレス値を代入する際に、プログラムモジュールを識別するための識別番号と、プログラムモジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグとを設定するための設定プログラムを生成する。

【０３３９】

再計算プログラム生成部２３ｋは、メモリ１９がコンパクションされた後、識別番号及び識別子に基づいてアドレス値を再計算するための再計算プログラムを生成する。

【０３４０】

端末装置１２は、プログラム実行部１８、メモリ１９及び通信部２１を備えて構成される。プログラム実行部１８は、識別番号及び識別フラグ設定部２４ｊ、コンパクション部２４ｂ及び再計算部２４ｋを備えて構成される。

【０３４１】

メモリ１９は、アドレス値を格納するアドレス値用メモリ領域と、数値等のアドレス値以外を格納する非アドレス値用メモリ領域に分割されている。また、メモリ１９には、プログラムモジュールを識別するための識別番号と、プログラムモジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグとを格納する領域が設けられている。

【０３４２】

識別番号及び識別フラグ設定部２４ｊは、プログラムモジュールの実行時において、ポインタにアドレス値を代入する際に、プログラムモジュールを識別するための識別番号と、プログラムモジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグとを設定する。

【０３４３】

再計算部２４ｋは、メモリ１９がコンパクションされた後、識別番号及び識別子に基づいてアドレス値を再計算する。

【０３４４】

まず、本変形例では、プログラムモジュールがメモリに記憶される際に、当該プログラムモジュールを識別する識別番号が付与され、この識別番号と当該プログラムモジュールのデータ領域の先頭アドレス及びコード領域の先頭アドレスとが対応付けられたテーブルが予め記憶される。そして、識別番号及び識別フラグ設定部２４ｊによって、プログラムモジュールの実行時において、ポインタにアドレス値を代入する際に、プログラムモジュールを識別するための識別番号と、プログラムモジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグとが設定される。なお、ポインタに対して、プログラムモジュールのデータ領域へのアドレスを設定する場合、識別フラグには１が設定され、プログラムモジュールのコード領域へのアドレスを設定する場合、識別フラグには０が設定される。そして、コンパクション部２４ｂによって、コンパクションが行われる。

【０３４５】

メモリ１９がコンパクションされた後、再計算部２４ｋによって、識別番号及び識別子に基づいて、予め記憶されているテーブルを参照し、アドレス値が再計算され、再計算されたアドレス値が新たなアドレス値として設定される。

【０３４６】

なお、本変形例では、メモリ１９を、アドレス値を格納するアドレス値用メモリ領域と、アドレス値以外を格納する非アドレス値用メモリ領域とに分割するとしているが、本発明は特にこれに限定されず、アドレス値の場所を示す情報を保持してもよい。

【０３４７】

このように、メモリ１９内において、モジュールを識別するための識別番号と、モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグとを設け、識別番号及び識別フラグ設定プログラム生成部２３ｊによって、ポインタにアドレス値を代入する際に識別番号と識別フラグとを設定するための設定プログラムが生成される。コンパクションプログラム生成部２３ｂによって、メモリ１９をコンパクションするためのコンパクションプログラムが生成される。メモリ１９がコンパクションされた後、再計算プログラム生成部２３ｋによって、識別番号及び識別フラグに基づいてアドレス値を再計算するための再計算プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【０３４８】

次に、メモリのフラグメンテーションを解消する第５の変形例について説明する。第５の変形例では、メモリ内の領域を、アドレス値が格納されるアドレス値用メモリ領域と、数値等のアドレス値以外が格納される非アドレス値用メモリ領域とに分割し、メモリ内のポインタ変数に対して、モジュールを識別するための識別番号と、モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグ（識別子）とを設け、モジュールの実行時において、ポインタ変数にアドレス値を代入する際に、識別番号と識別フラグとを設定し、当該アドレス値には、データ領域の先頭アドレスからのオフセット値又はコード領域の先頭アドレスからのオフセット値を設定する。なお、プログラムモジュールの実行時に、ポインタ変数を使用する際は、データ領域の先頭アドレス又はコード領域の先頭アドレスにオフセット値を加算することによって実アドレスに変換して使用する。

【０３４９】

なお、本変形例と上述の第４の変形例とは、アドレス値をオフセット値として記憶する点のみが異なるため、説明を省略する。

【０３５０】

このように、メモリ１９内において、モジュールを識別するための識別番号と、モジュールのデータ領域へのアドレスを設定するか、コード領域へのアドレスを設定するかを識別する識別フラグとを設け、ポインタにアドレス値を代入する際に識別番号と識別フラグとを設定するとともに、データ領域の先頭アドレス又はコード領域の先頭アドレスからの

オフセット値を設定するためのオフセット値設定プログラムが生成される。そして、ポインタを使用する際に、オフセット値にデータ領域の先頭アドレス又はコード領域の先頭アドレスを加算することによって実アドレスに変換するための実アドレス変換プログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【０３５１】

次に、メモリのフラグメンテーションを解消する第６の変形例について説明する。第６の変形例では、リファレンスがオブジェクトを間接的に参照するハンドルを設け、コンパクションによって移動するコード領域又はデータ領域を指すハンドルがあるか否かを検索し、コード領域又はデータ領域を指すハンドルが発見された場合、当該ハンドルを再配置し、全てのハンドルについて再配置を行った後、当該コード領域又はデータ領域をコンパクションするものである。

【０３５２】

図２７は、第６の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。なお、他の実施の形態と同じ構成物については、同じ符号を付し、説明を省略する。

【０３５３】

図２７に示すように、サーバ１１は、通信部２２及びコンパイル装置１４を備えて構成される。コンパイル装置１４は、プログラム実行部１６及びプログラム記憶部１７を備えて構成される。プログラム実行部１６は、ハンドル検索プログラム生成部２３１、コンパクションプログラム生成部２３ｂ及びハンドル再配置プログラム生成部２３ｍを備えて構成される。

【０３５４】

ハンドル検索プログラム生成部２３１は、コンパクションによって移動するプログラムモジュールのコード領域又はデータ領域を指すハンドルがあるか否かを検索するための検索プログラムを生成する。

【０３５５】

コンパクションプログラム生成部２３ｂは、プログラムモジュールのコード領域又はデータ領域毎にコンパクションするためのコンパクションプログラムを生成する。

【０３５６】

ハンドル再配置プログラム生成部２３ｍは、コード領域又はデータ領域を指すハンドルが発見された場合、当該ハンドルを再配置するためのハンドル再配置プログラムを生成する。

【０３５７】

また、コンパクションプログラム生成部２３ｂは、全てのハンドルについて再配置を行った後、当該コード領域又はデータ領域をコンパクションするためのコンパクションプログラムを生成する。

【０３５８】

端末装置１２は、プログラム実行部１８、メモリ１９及び通信部２１を備えて構成される。プログラム実行部１８は、ハンドル検索部２４１、コンパクション部２４ｂ及びハンドル再配置部２４ｍを備えて構成される。

【０３５９】

メモリ１９には、リファレンスがオブレクトのアドレスを間接的に参照するためのハンドルのための領域が設けられている。したがって、リファレンスは、オブジェクトを直接的に参照するのではなく、ハンドルを介して間接的に参照する。ハンドルとオブジェクトとの関係は一対一に対応している。これは、オブジェクトを指すハンドルが複数存在しないことを意味している。また、リファレンスとオブジェクトとの関係は多対一に対応している。オブジェクトは、複数の異なるリファレンスから参照される可能性があるためである。

【０３６０】

ハンドル検索部２４１は、コンパクションによって移動するプログラムモジュールのコード領域又はデータ領域を指すハンドルがあるか否かを検索する。コンパクション部２４ｂは、プログラムモジュールのコード領域又はデータ領域毎にコンパクションする。ハンドル再配置部２４ｍは、コード領域又はデータ領域を指すハンドルが発見された場合、当該ハンドルのアドレス値を再配置する。また、コンパクション部２４ｂは、全てのハンドルについて再配置を行った後、当該コード領域又はデータ領域をコンパクションする。

【０３６１】

このように、ハンドル検索プログラム生成部２３１によって、コンパクションによって移動するコード領域又はデータ領域を指すハンドルがあるか否かを検索するための検索プログラムが生成される。そして、コード領域又はデータ領域を指すハンドルが発見された場合、ハンドル再配置プログラム生成部２３ｍによって、当該ハンドルを再配置するためのハンドル再配置プログラムが生成される。全てのハンドルについて再配置を行った後、コンパクションプログラム生成部２３ｂによって、当該コード領域又はデータ領域をコンパクションするためのコンパクションプログラムが生成される。したがって、メモリのフラグメンテーションを解消することができ、メモリを有効的に利用することができる。

【０３６２】

また以上説明したような原始プログラム、目的プログラム、その他のプログラムの全てはコンピュータ読み取り可能な記録媒体に格納されて、コンピュータから読み出されて利用又は実行され、場合によってはインターネット等のネットワークを介して他のコンピュータと通信される。同様に、このような翻訳を行うコンパイラプログラムもまたコンピュータ読み取り可能な記録媒体に格納されて、コンピュータから読み出されてコンパイル（原始プログラムから目的プログラムへの翻訳・変換）が実行され、場合によってはインターネット等のネットワークを介して他のコンピュータと通信される。

【０３６３】

このコンパイラプログラムによってコンピュータが行うのはこのコンパイラプログラムによって規定されたコンパイル方法であり、このコンピュータハードウエアとコンパイラプログラムとは全体としてコンパイラ装置を構成する。

【産業上の利用可能性】

【０３６４】

本発明に係るコンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置は、自動的に再入可能な目的プログラムを生成することができ、あるプログラム言語で記述された原始プログラムを、あるコンピュータによって実行するための目的プログラムに変換するためのコンパイラプログラム、コンパイラプログラムを記録したコンピュータ読み取り可能な記録媒体、コンパイル方法及びコンパイル装置等として有用である。

【図面の簡単な説明】

【０３６５】

【図１】本発明の一実施の形態による動的モジュールリンクシステムの構成を示す図である。

【図２】図１に示すコンパイル装置の構成を示すブロック図である。

【図３】図２に示すコンパイル装置によるコンパイラ処理の一例を示すフローチャートである。

【図４】本発明の第１の実施の形態におけるコンパイル装置が変換するソースプログラムと、コンパイル装置によって変換されたオブジェクトプログラムとの構造を示す図である。

【図５】図１に示す端末装置の構成を示すブロック図である。

【図６】図５に示す端末装置によってアプリケーションプログラムモジュールがライブラリプログラムモジュールを呼び出す処理を示すフローチャートである。

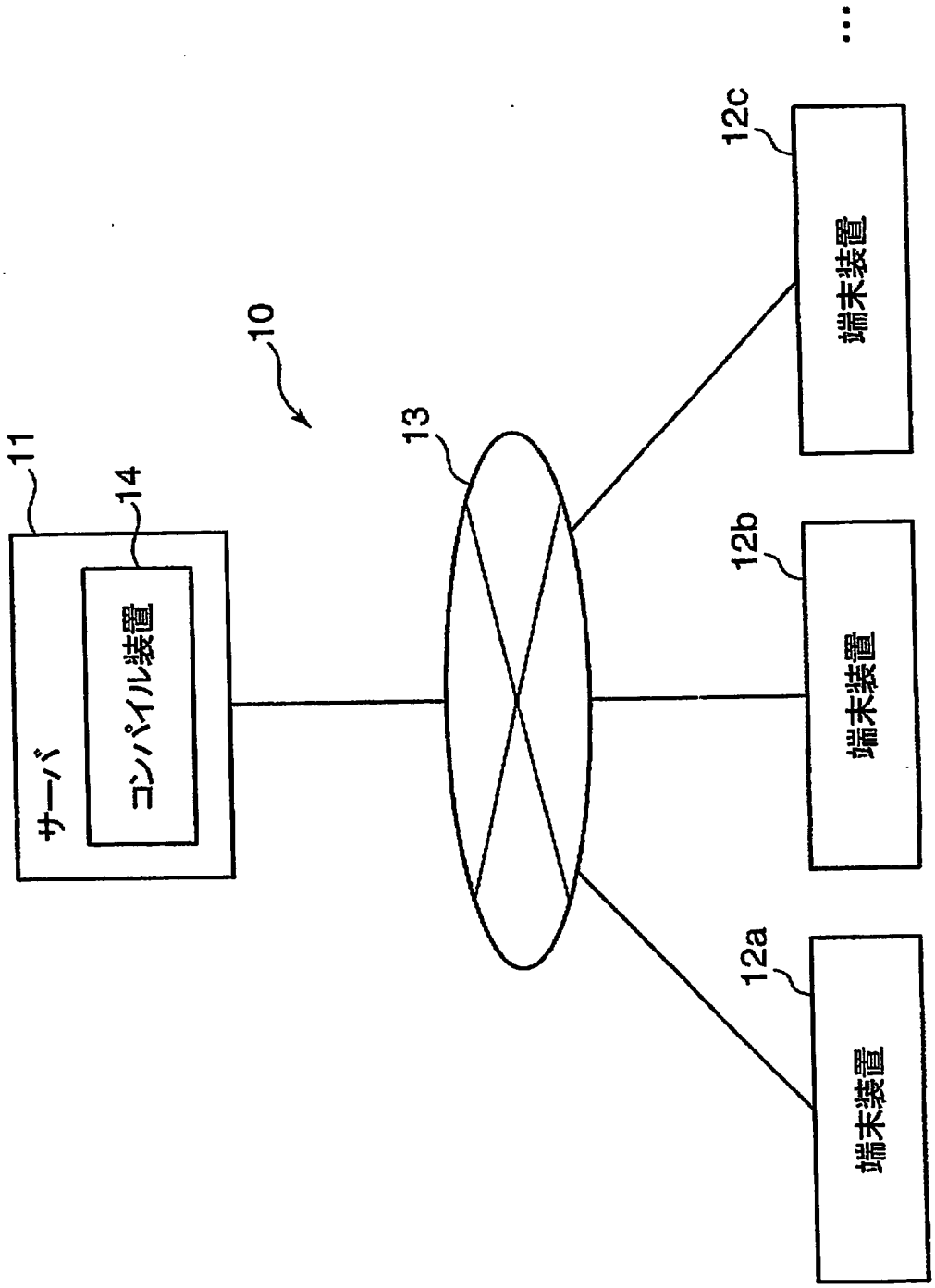
【図７】アプリケーションプログラムモジュールからライブラリプログラムモジュールを呼び出す処理を説明するための図である。

【図８】アプリケーションプログラムモジュール及びライブラリプログラムモジュールのソースプログラム及びオブジェクトプログラムの一例を示す図である。
【図９】スタックの状態を説明するための図である。
【図１０】第２の実施の形態におけるソースプログラム及びオブジェクトプログラムの一例を示す図である。
【図１１】本発明の第２の実施の形態におけるコンパイル装置の構成を示すブロック図である。
【図１２】図１１に示すコンパイル装置によるコンパイラ処理の一例を示すフローチャートである。
【図１３】本発明の第２の実施の形態におけるライブラリプログラムモジュールのコード領域及びデータ領域を示す図である。
【図１４】本発明の第３の実施の形態におけるライブラリプログラムモジュールのソースプログラム及びオブジェクトプログラムの一例を示す図である。
【図１５】本発明の第３の実施の形態におけるライブラリプログラムモジュールのコード領域及びデータ領域を示す図である。
【図１６】本発明の第４の実施の形態であるコンパイラが変換の対象とする呼び出し元プログラムであるアプリケーションプログラムモジュールと、呼び出されるプログラムであるライブラリプログラムモジュールと、その呼び出し処理時に行われる処理の一例を概念的に示す図である。
【図１７】本発明の第５の実施の形態におけるコンパイル装置の構成を示すブロック図である。
【図１８】本発明の第５の実施の形態における端末装置の構成を示すブロック図である。
【図１９】図１８に示す端末装置による動的モジュールリンク処理の一例を示すフローチャートである。
【図２０】フラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。
【図２１】第１の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。
【図２２】第２の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。
【図２３】第２の変形例におけるフラグメンテーションの解消方法について説明するための図である。
【図２４】第３の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。
【図２５】第３の変形例におけるフラグメンテーションの解消方法について説明するための図である。
【図２６】第４の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。
【図２７】第６の変形例におけるフラグメンテーションを解消するためのコンパイル装置及び端末装置の構成を示すブロック図である。

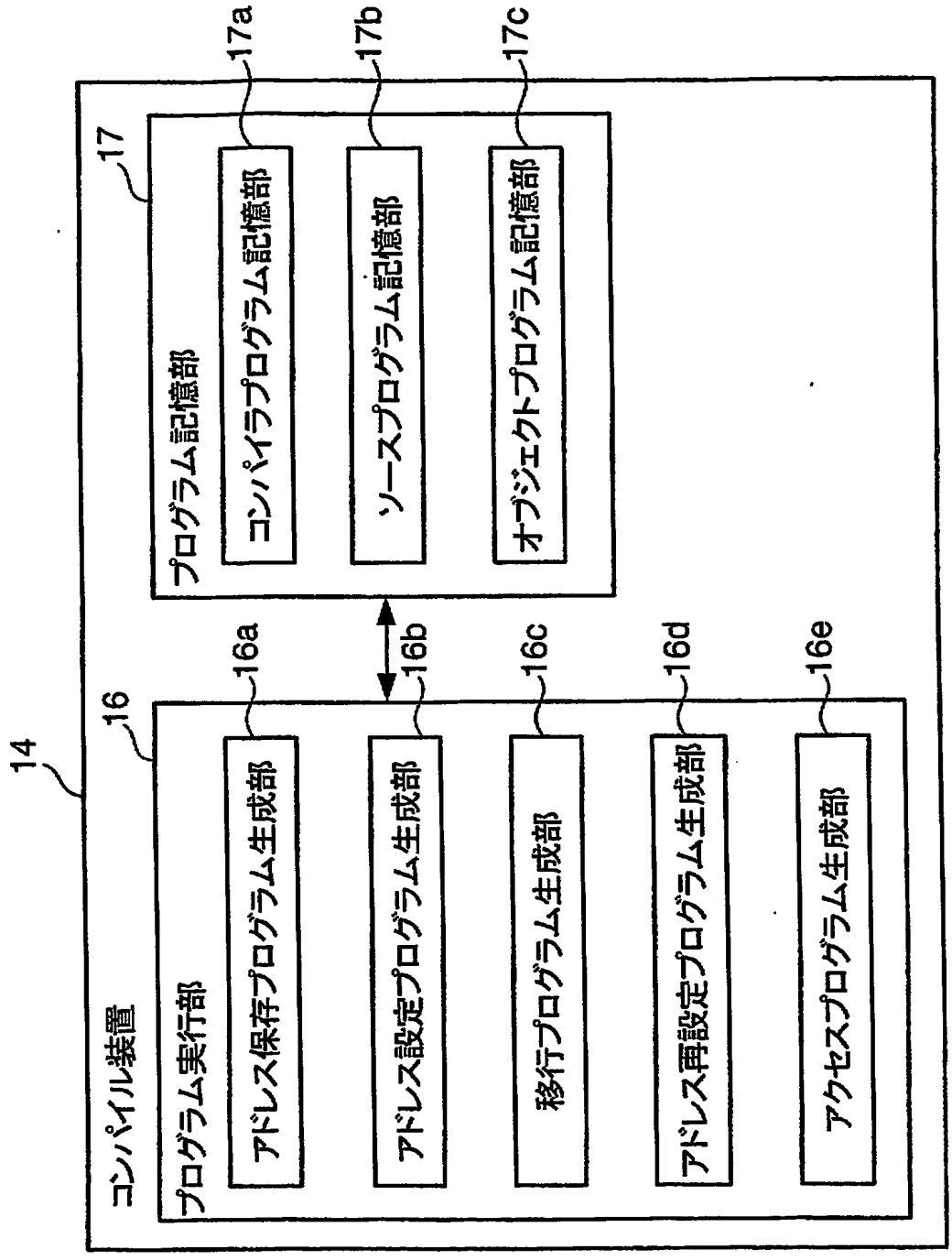
【符号の説明】
【０３６６】
１０　動的モジュールリンクシステム
１１　サーバ
１２　端末装置
１３　インターネット
１４　コンパイル装置
１６，１８　プログラム実行部
１６ａ　アドレス保存プログラム生成部

１６ｂ　アドレス設定プログラム生成部
１６ｃ　移行プログラム生成部
１６ｄ　アドレス再設定プログラム生成部
１６ｅ　アクセスプログラム生成部
１７　プログラム記憶部
１７ａ　コンパイラプログラム記憶部
１７ｂ　ソースプログラム記憶部
１７ｃ　オブジェクトプログラム記憶部
１８ａ　アドレス保存部
１８ｂ　アドレス設定部
１８ｃ　移行部
１８ｄ　アドレス再設定部
１８ｅ　アクセス部
１９　メモリ
１９ａ　アプリケーションプログラムモジュール記憶部
１９ｂ　ライブラリプログラムモジュール記憶部
１９ｃ　スタック

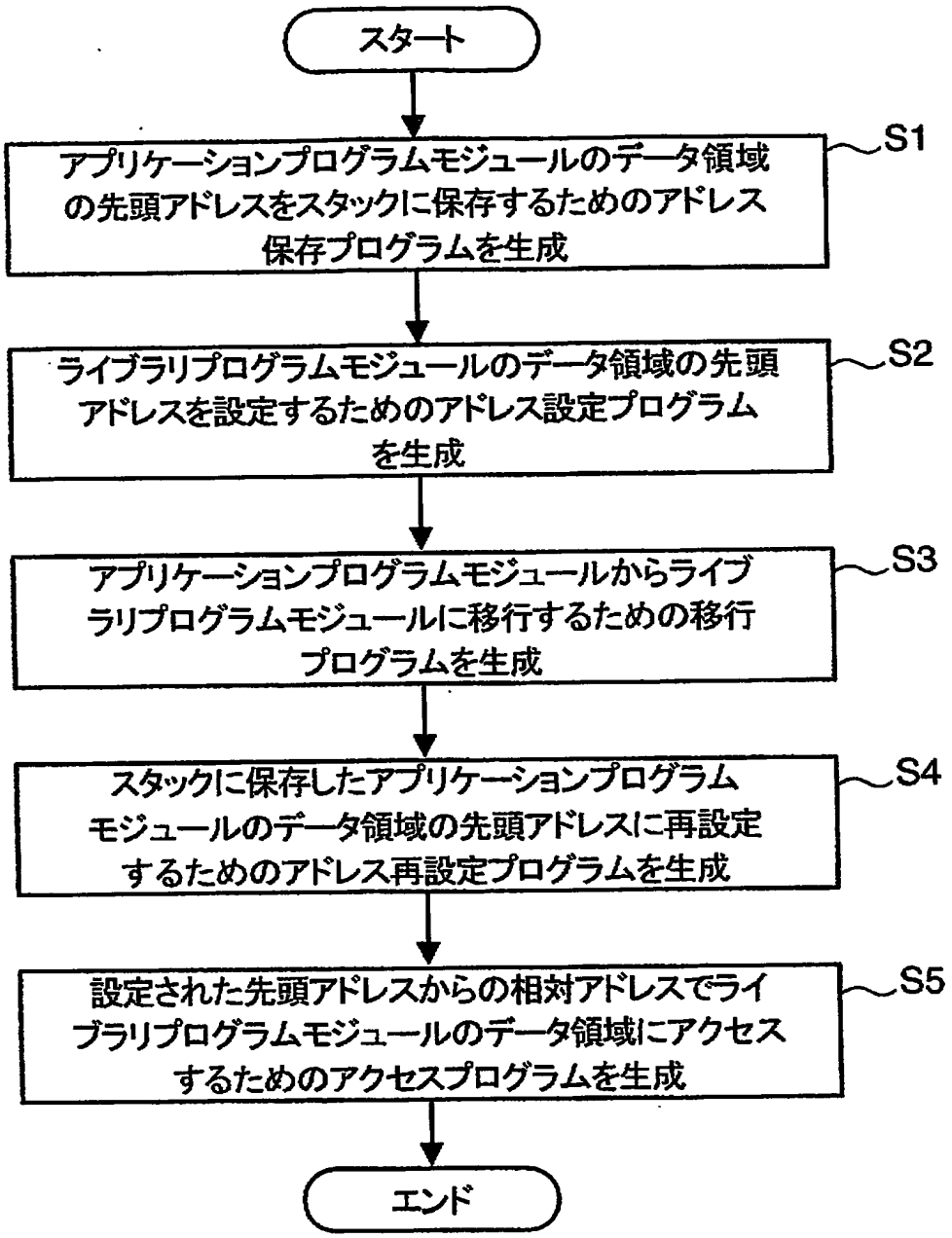
【書類名】図面
【図１】



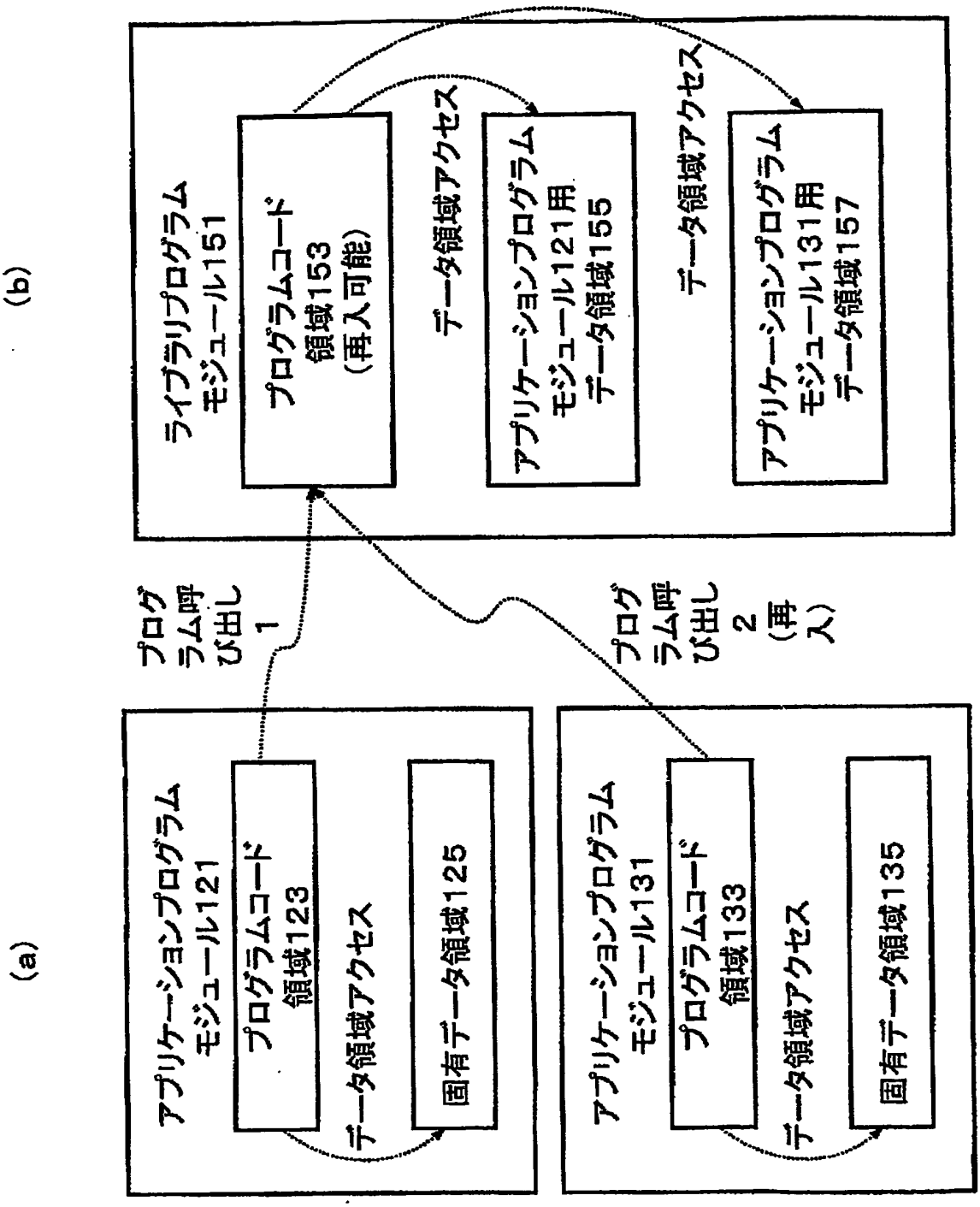
【図２】


14
コンパイル装置
16
プログラム実行部
16a
アドレス保存プログラム生成部
16b
アドレス設定プログラム生成部
16c
移行プログラム生成部
16d
アドレス再設定プログラム生成部
16e
アクセスプログラム生成部
17
プログラム記憶部
17a
コンパイラプログラム記憶部
17b
ソースプログラム記憶部
17c
オブジェクトプログラム記憶部

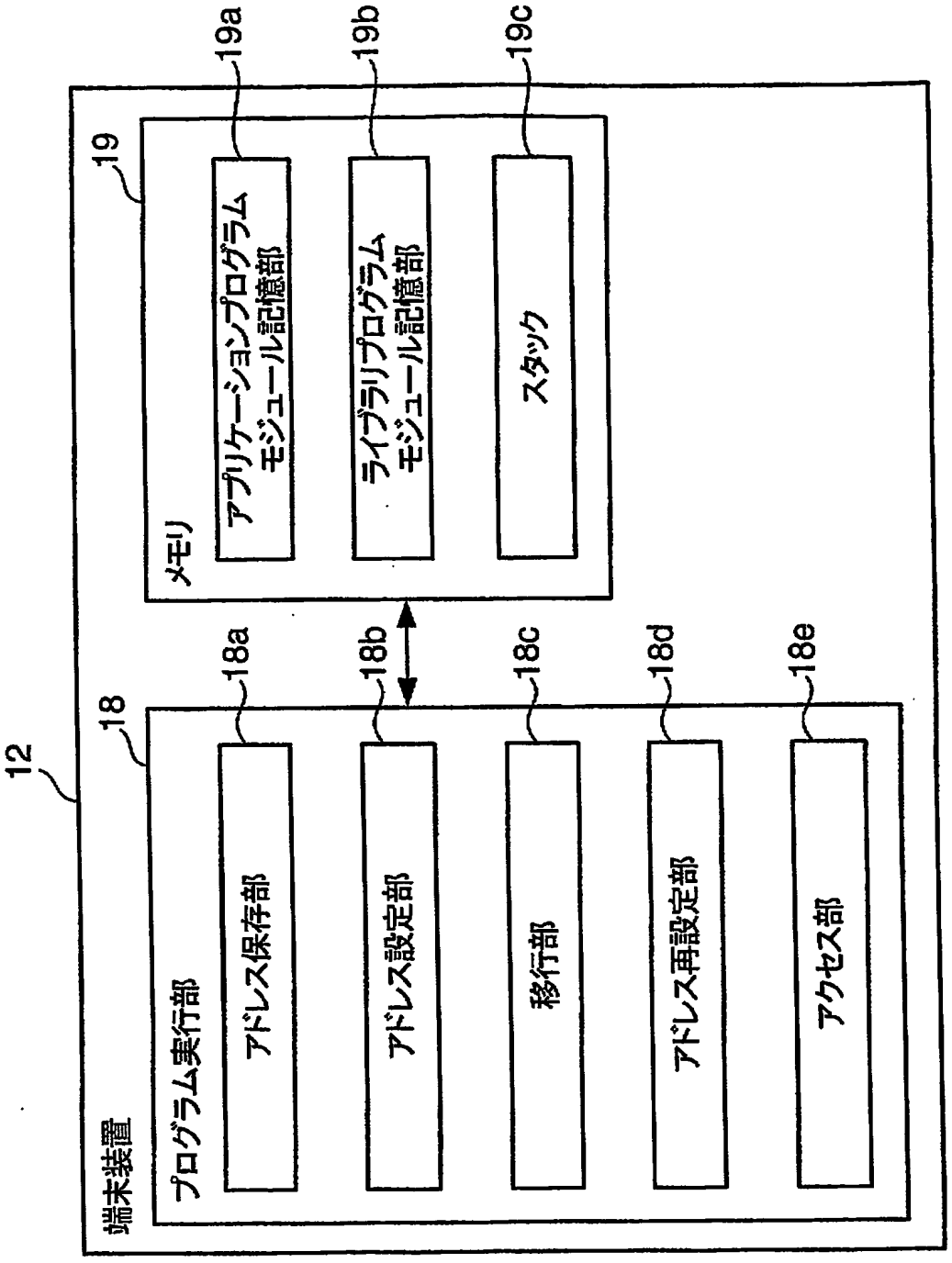
【図３】


スタート
アプリケーションプログラムモジュールのデータ領域の先頭アドレスをスタックに保存するためのアドレス保存プログラムを生成
S1
ライブラリプログラムモジュールのデータ領域の先頭アドレスを設定するためのアドレス設定プログラムを生成
S2
アプリケーションプログラムモジュールからライブラリプログラムモジュールに移行するための移行プログラムを生成
S3
スタックに保存したアプリケーションプログラムモジュールのデータ領域の先頭アドレスに再設定するためのアドレス再設定プログラムを生成
S4
設定された先頭アドレスからの相対アドレスでライブラリプログラムモジュールのデータ領域にアクセスするためのアクセスプログラムを生成
S5
エンド

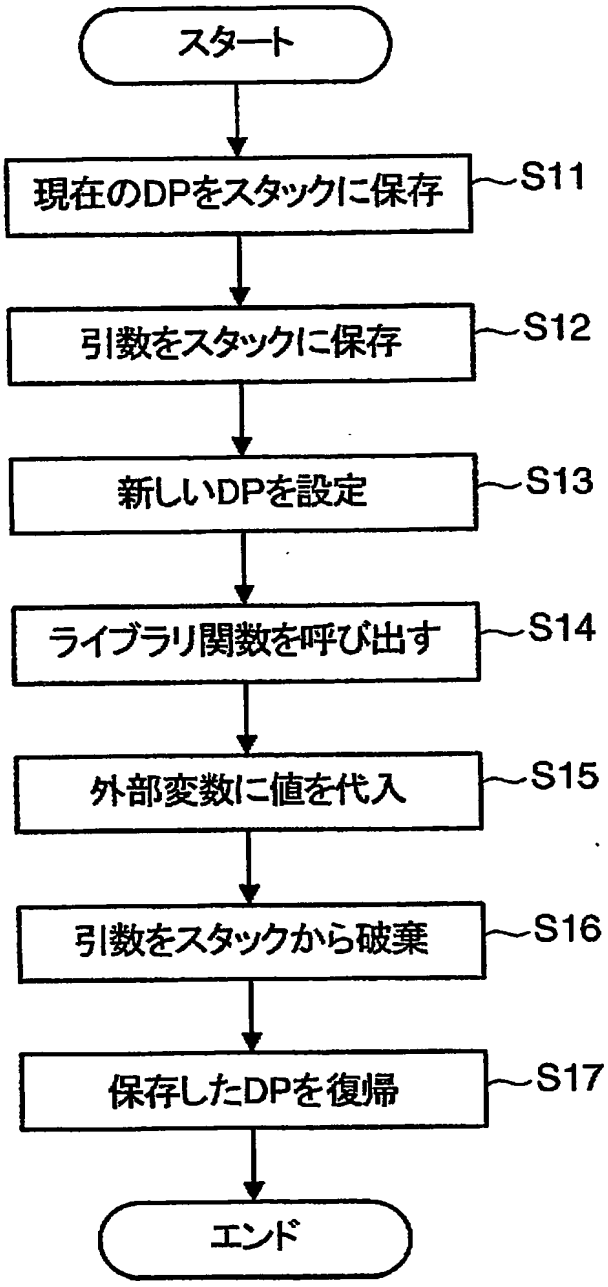
【図４】


(a)
アプリケーションプログラム
モジュール121
プログラムコード
領域123
データ領域アクセス
固有データ領域125
アプリケーションプログラム
モジュール131
プログラムコード
領域133
データ領域アクセス
固有データ領域135
プログラム呼び出し 1
プログラム呼び出し 2 (再入)
(b)
ライブラリプログラム
モジュール151
プログラムコード
領域153
(再入可能)
データ領域アクセス
アプリケーションプログラム
モジュール121用
データ領域155
データ領域アクセス
アプリケーションプログラム
モジュール131用
データ領域157

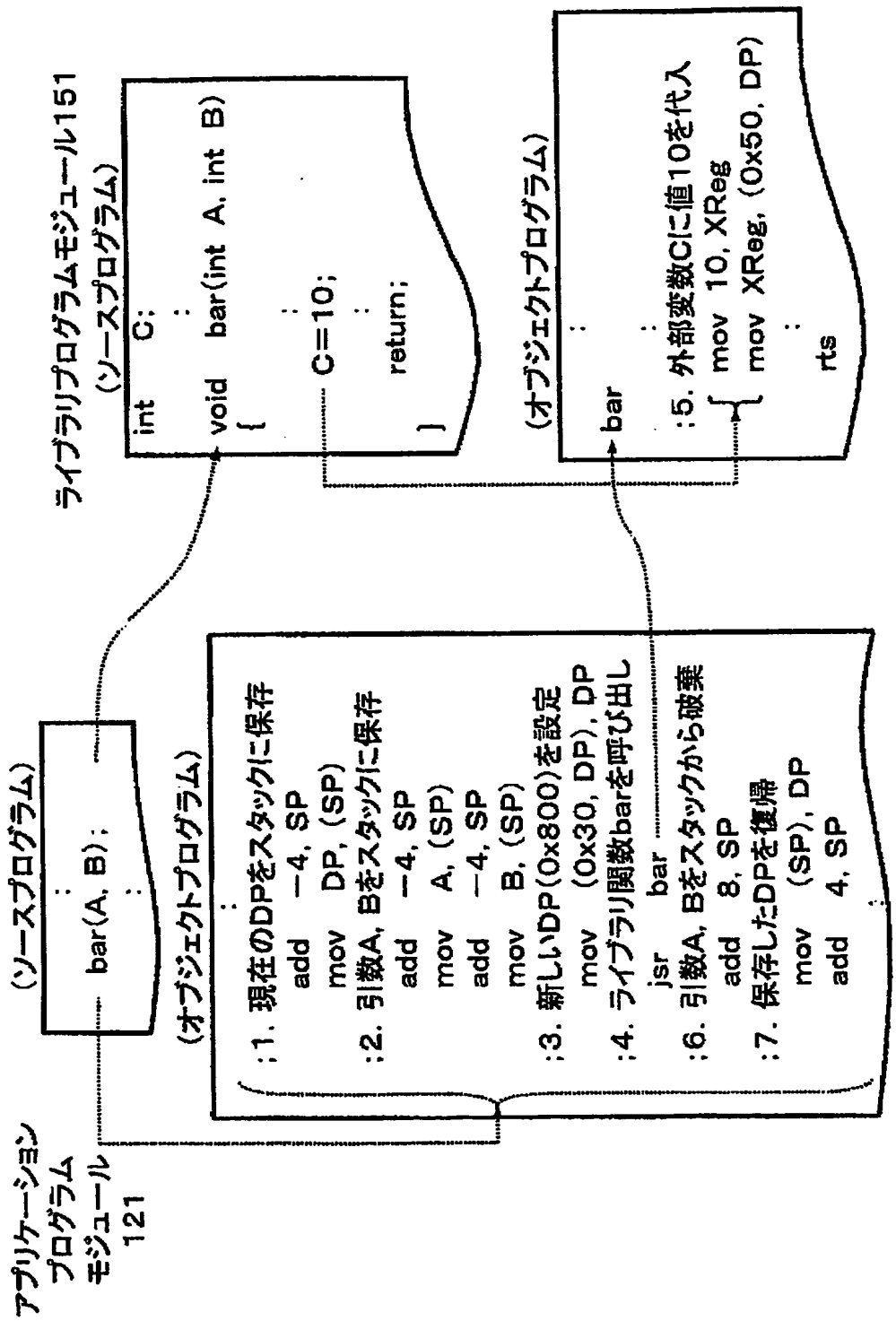
【図５】


12
端末装置
18
プログラム実行部
アドレス保存部
アドレス設定部
移行部
アドレス再設定部
アクセス部
18a
18b
18c
18d
18e
19
メモリ
アプリケーションプログラム
モジュール記憶部
ライブラリプログラム
モジュール記憶部
スタック
19a
19b
19c

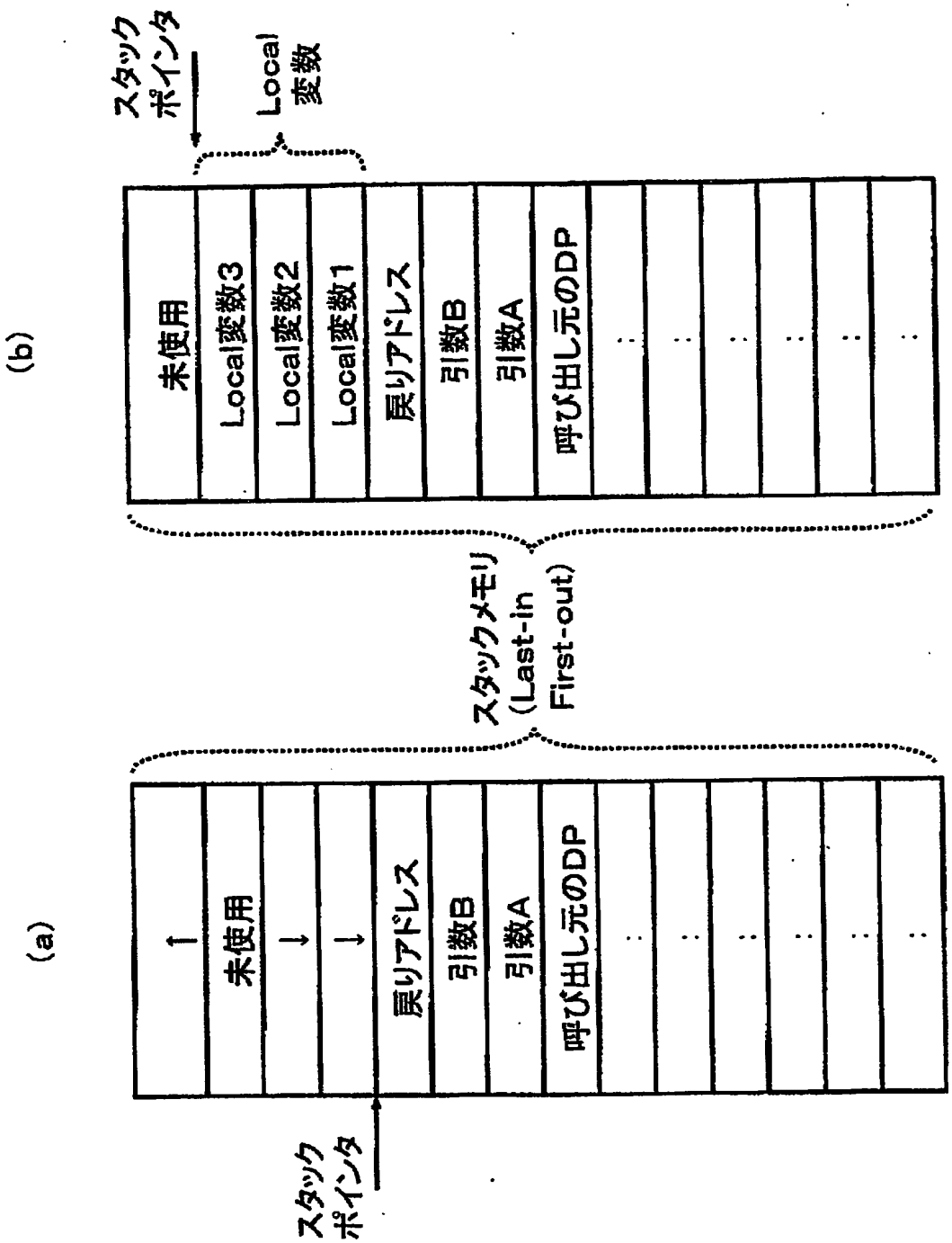
【図６】


スタート
現在のDPをスタックに保存
S11
引数をスタックに保存
S12
新しいDPを設定
S13
ライブラリ関数を呼び出す
S14
外部変数に値を代入
S15
引数をスタックから破棄
S16
保存したDPを復帰
S17
エンド

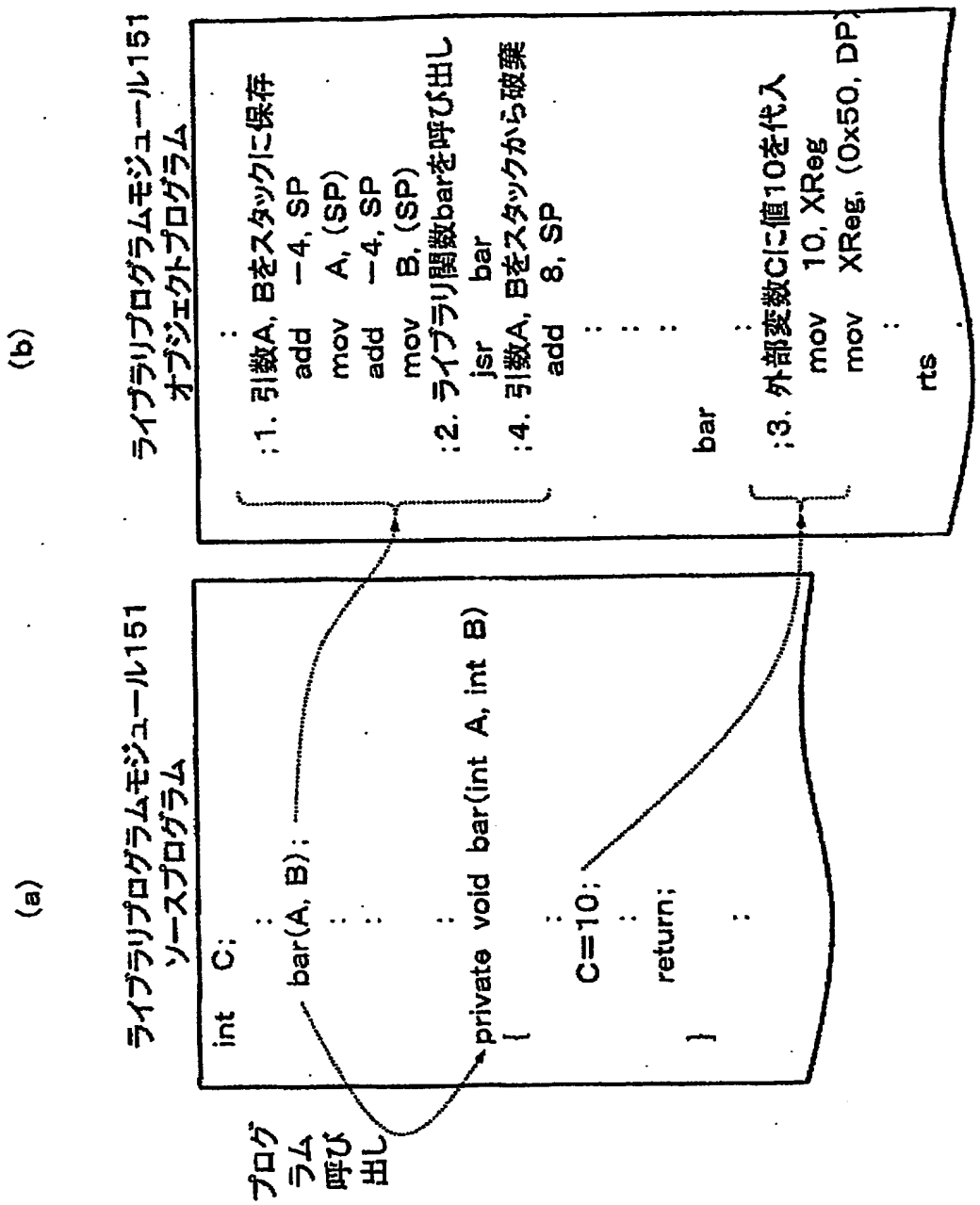
【図８】


(a)
アプリケーション
プログラム
モジュール
121
(ソースプログラム)
bar(A, B);
(オブジェクトプログラム)
:1. 現在のDPをスタックに保存
add −4, SP
mov DP, (SP)
:2. 引数A, Bをスタックに保存
add −4, SP
mov A, (SP)
add −4, SP
mov B, (SP)
:3. 新しいDP(0x800)を設定
mov (0x30, DP), DP
:4. ライブラリ関数barを呼び出し
jsr bar
:6. 引数A, Bをスタックから破棄
add 8, SP
:7. 保存したDPを復帰
mov (SP), DP
add 4, SP
(b)
ライブラリプログラムモジュール151
(ソースプログラム)
int C;
void bar(int A, int B)
{
C=10;
return;
}
(オブジェクトプログラム)
bar
:5. 外部変数Cに値10を代入
mov 10, XReg
mov XReg, (0x50, DP)
rts

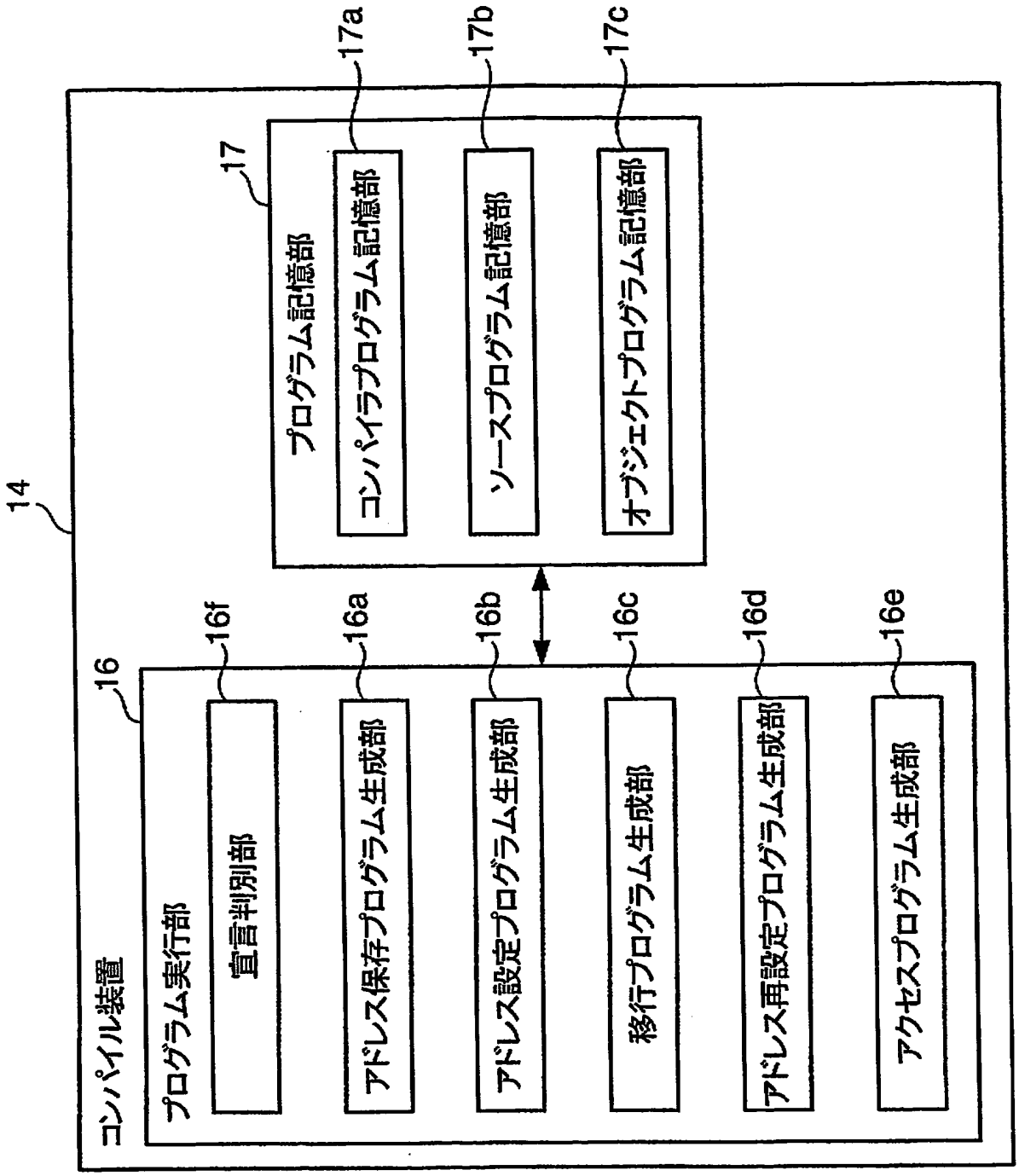
【図９】


(a)
(b)
スタックメモリ
(Last-in
First-out)
スタック
ポインタ
スタック
ポインタ
Local
変数
↑
未使用
→
→
戻りアドレス
引数B
引数A
呼び出し元のDP
未使用
Local変数3
Local変数2
Local変数1
戻りアドレス
引数B
引数A
呼び出し元のDP

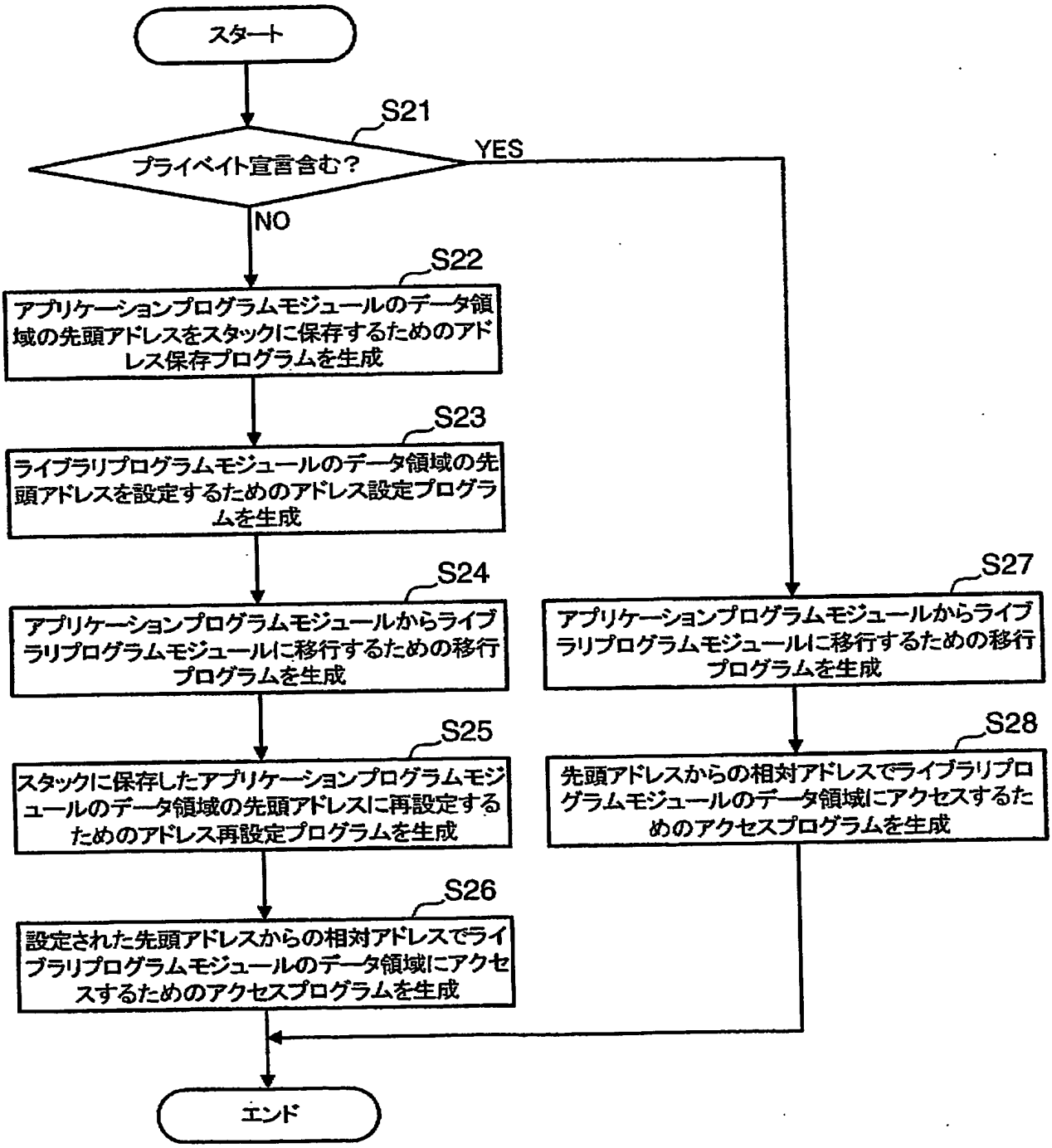
【図１０】


(a)
ライブラリプログラムモジュール151
ソースプログラム
int C;
bar(A, B);
private void bar(int A, int B)
{
C=10;
return;
}
プログラム呼び出し
(b)
ライブラリプログラムモジュール151
オブジェクトプログラム
:1. 引数A, Bをスタックに保存
add -4, SP
mov A, (SP)
add -4, SP
mov B, (SP)
:2. ライブラリ関数barを呼び出し
jsr bar
:4. 引数A, Bをスタックから破棄
add 8, SP
bar
:3. 外部変数Cに値10を代入
mov 10, XReg
mov XReg, (0x50, DP)
rts

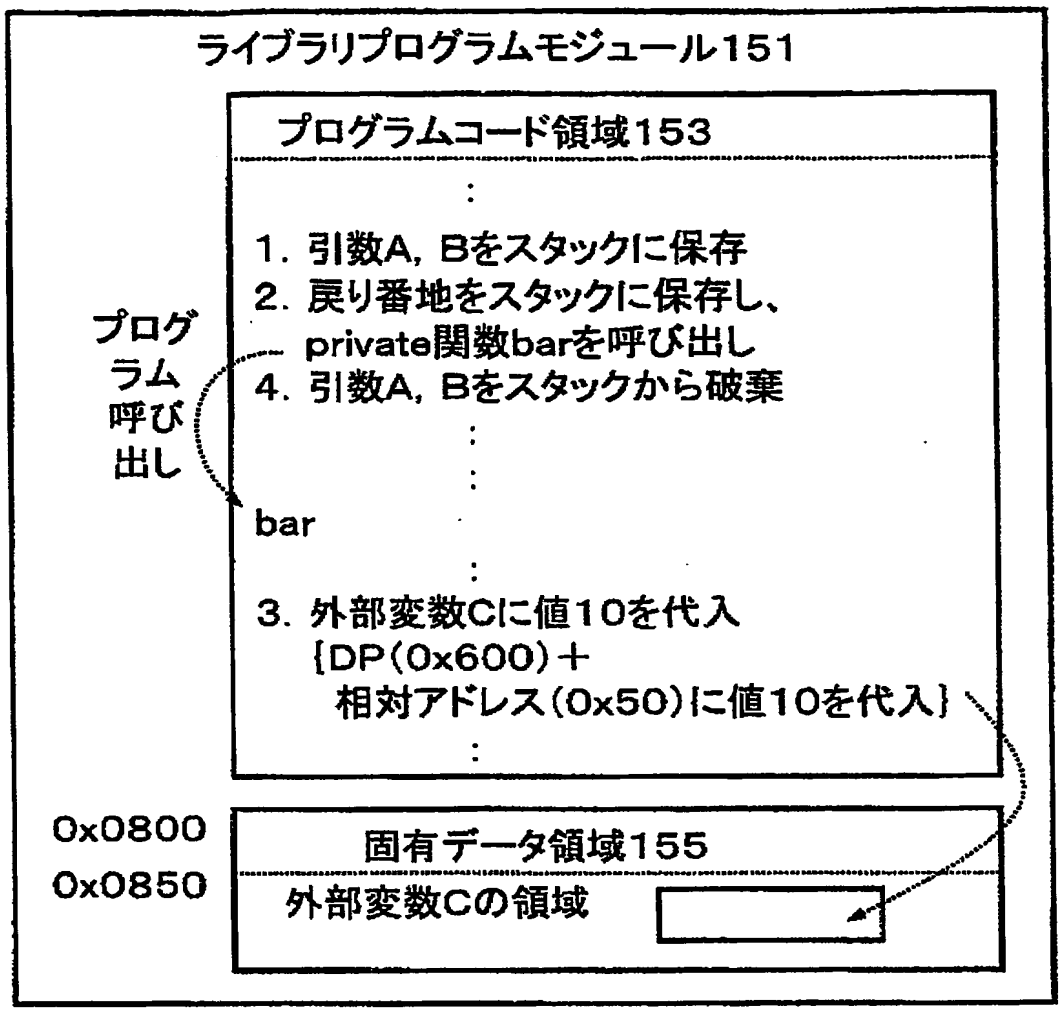
【図１１】


14
コンパイル装置
16
プログラム実行部
宣言判別部
16f
アドレス保存プログラム生成部
16a
アドレス設定プログラム生成部
16b
移行プログラム生成部
16c
アドレス再設定プログラム生成部
16d
アクセスプログラム生成部
16e
17
プログラム記憶部
コンパイラプログラム記憶部
17a
ソースプログラム記憶部
17b
オブジェクトプログラム記憶部
17c

【図１２】


スタート
S21
プライベイト宣言含む？
YES
NO
S22
アプリケーションプログラムモジュールのデータ領域の先頭アドレスをスタックに保存するためのアドレス保存プログラムを生成
S23
ライブラリプログラムモジュールのデータ領域の先頭アドレスを設定するためのアドレス設定プログラムを生成
S24
アプリケーションプログラムモジュールからライブラリプログラムモジュールに移行するための移行プログラムを生成
S25
スタックに保存したアプリケーションプログラムモジュールのデータ領域の先頭アドレスに再設定するためのアドレス再設定プログラムを生成
S26
設定された先頭アドレスからの相対アドレスでライブラリプログラムモジュールのデータ領域にアクセスするためのアクセスプログラムを生成
S27
アプリケーションプログラムモジュールからライブラリプログラムモジュールに移行するための移行プログラムを生成
S28
先頭アドレスからの相対アドレスでライブラリプログラムモジュールのデータ領域にアクセスするためのアクセスプログラムを生成
エンド

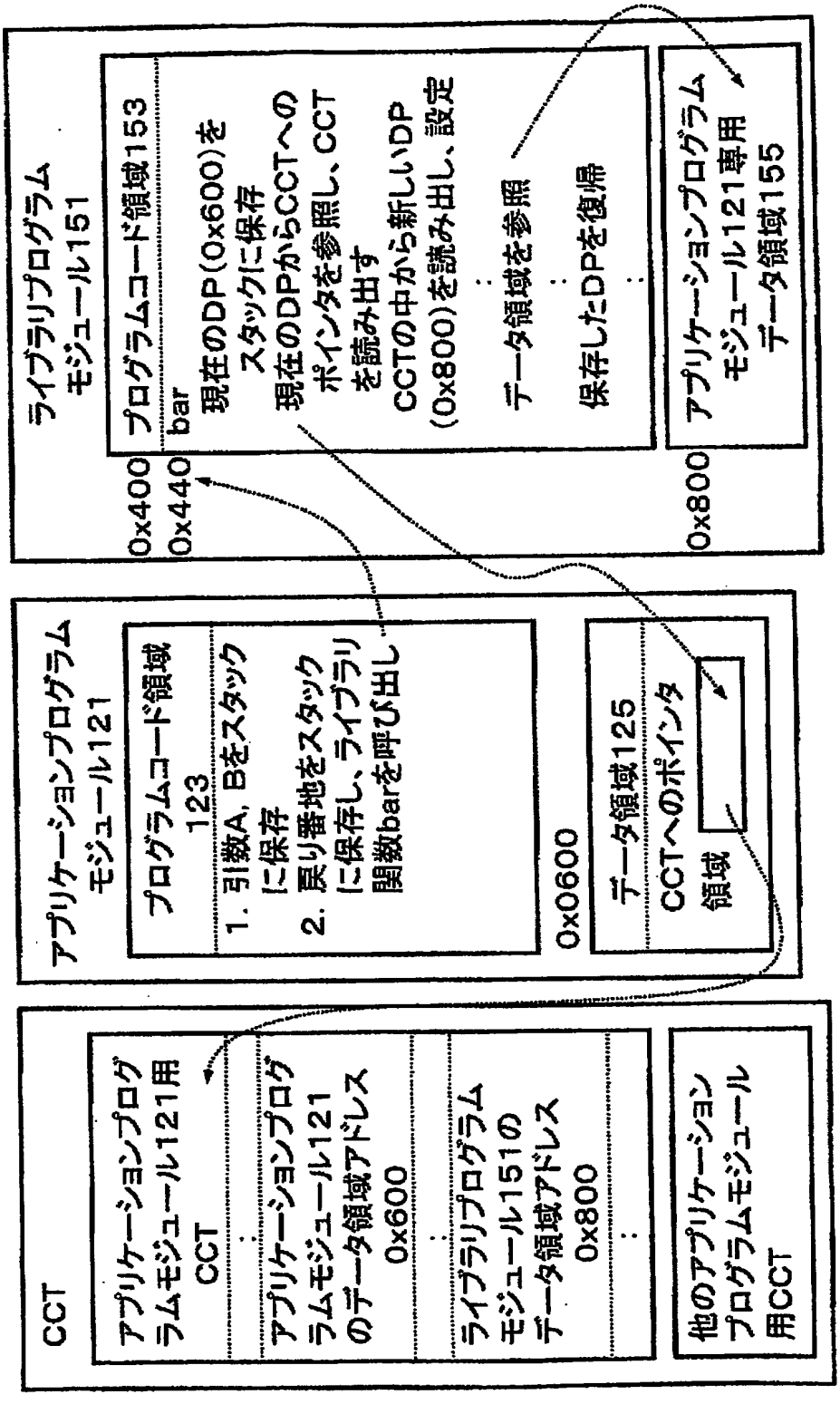
【図１３】


ライブラリプログラムモジュール151
プログラムコード領域153
1. 引数A, Bをスタックに保存
2. 戻り番地をスタックに保存し、
private関数barを呼び出し
4. 引数A, Bをスタックから破棄
プログラム呼び出し
bar
3. 外部変数Cに値10を代入
{DP(0x600)＋
相対アドレス(0x50)に値10を代入}
0x0800
0x0850
固有データ領域155
外部変数Cの領域

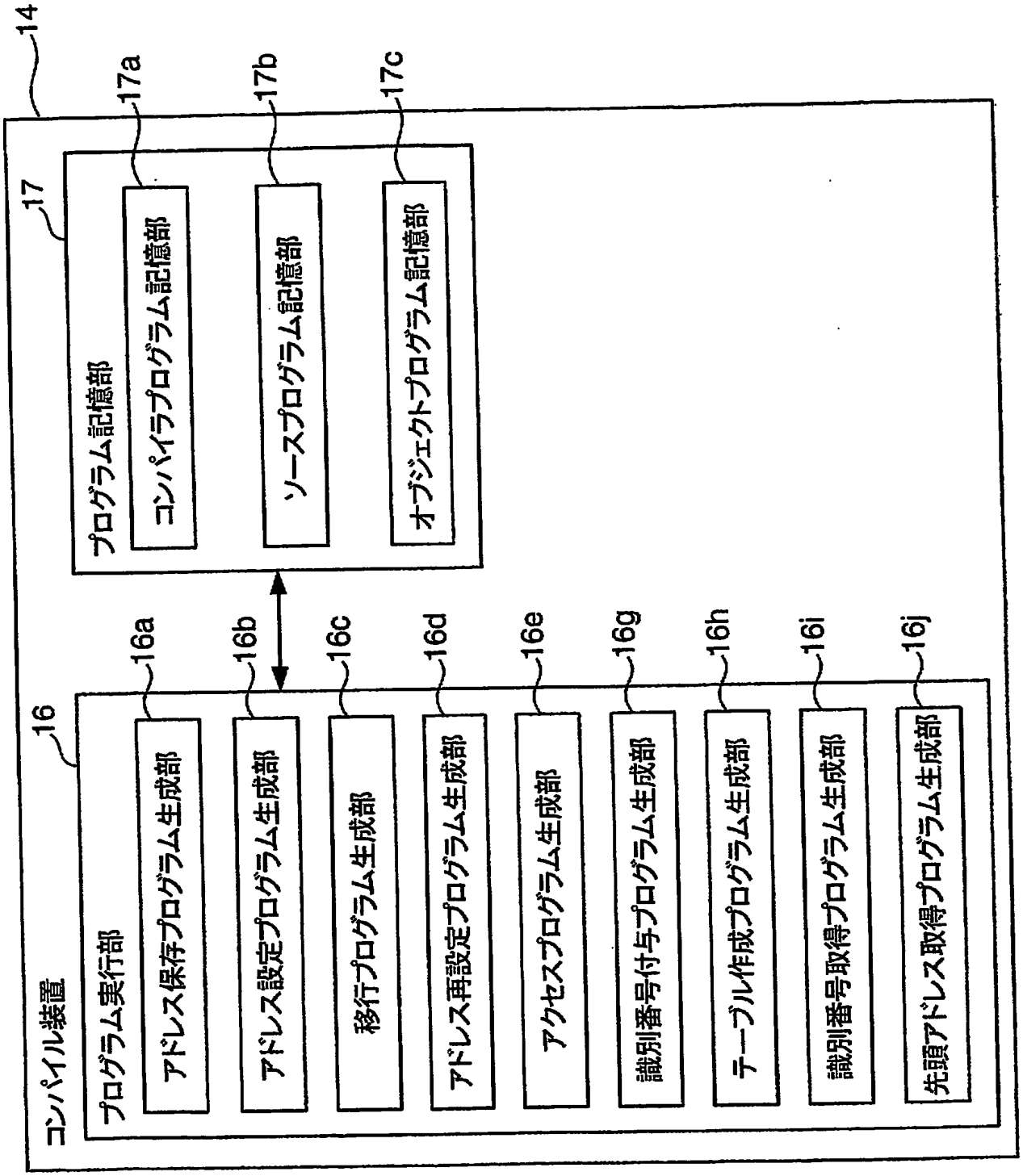
【図１４】

(a)
ライブラリプログラムモジュール551
(ソースプログラム)
int C;
common int D;
void bar(int A, int B)
{
/* 外部変数Cに値10を代入 */
C=10;
/* 外部変数Dに値20を代入 */
D=20;
return;
}
(b)
ライブラリプログラムモジュール551
(オブジェクトプログラム)
bar
:外部変数Cに値10を代入
mov 10, XReg
mov XReg, (0x50, DP)
:外部変数Dに値20を代入
mov 20, XReg
mov XReg, (0x60, SDP)
rts

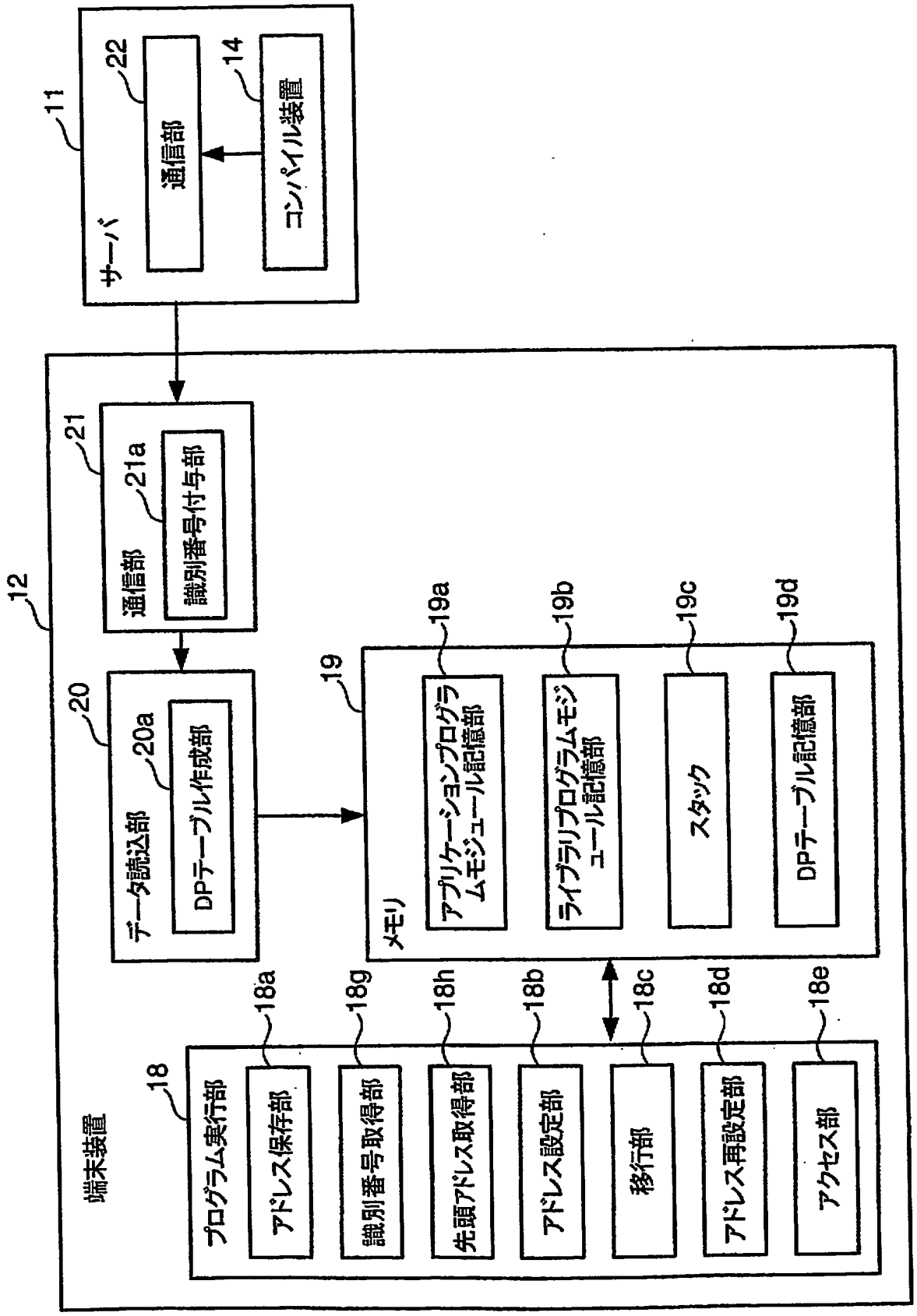
【図１６】


(a)
CCT
アプリケーションプログラムモジュール121用CCT
アプリケーションプログラムモジュール121のデータ領域アドレス 0x600
ライブラリプログラムモジュール151のデータ領域アドレス 0x800
他のアプリケーションプログラムモジュール用CCT
(b)
アプリケーションプログラムモジュール121
プログラムコード領域123
1. 引数A, Bをスタックに保存
2. 戻り番地をスタックに保存し、ライブラリ関数barを呼び出し
0x0600
データ領域125
CCTへのポインタ領域
(c)
ライブラリプログラムモジュール151
0x400
0x440
プログラムコード領域153
bar
現在のDP(0x600)をスタックに保存
現在のDPからCCTへのポインタを参照し、CCTを読み出す
CCTの中から新しいDP(0x800)を読み出し、設定
データ領域を参照
保存したDPを復帰
0x800
アプリケーションプログラムモジュール121専用データ領域155

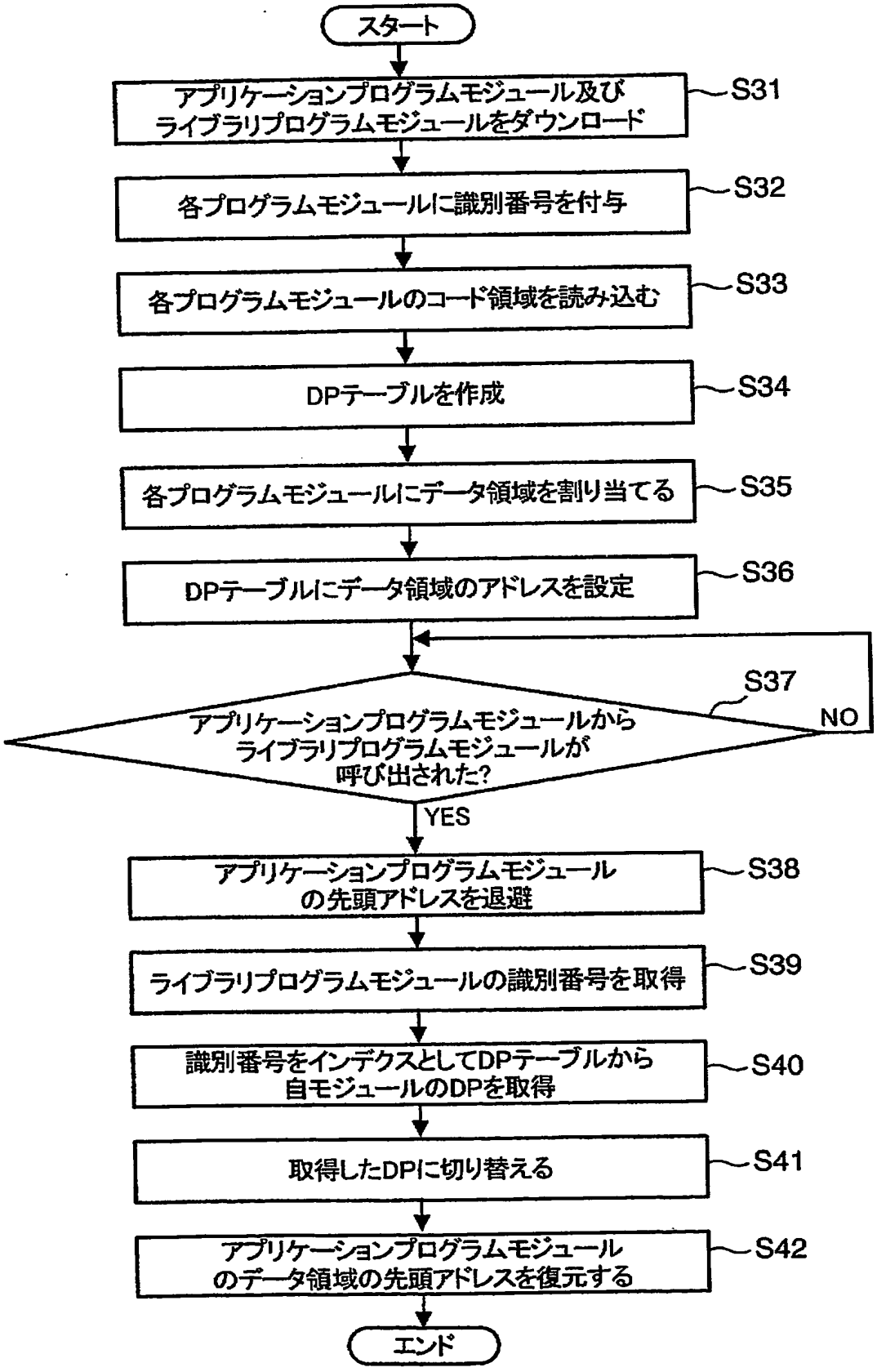
【図１７】


14
17
17a
17b
17c
プログラム記憶部
コンパイラプログラム記憶部
ソースプログラム記憶部
オブジェクトプログラム記憶部
16
コンパイル装置
16a
16b
16c
16d
16e
16g
16h
16i
16j
プログラム実行部
アドレス保存プログラム生成部
アドレス設定プログラム生成部
移行プログラム生成部
アドレス再設定プログラム生成部
アクセスプログラム生成部
識別番号付与プログラム生成部
テーブル作成プログラム生成部
識別番号取得プログラム生成部
先頭アドレス取得プログラム生成部

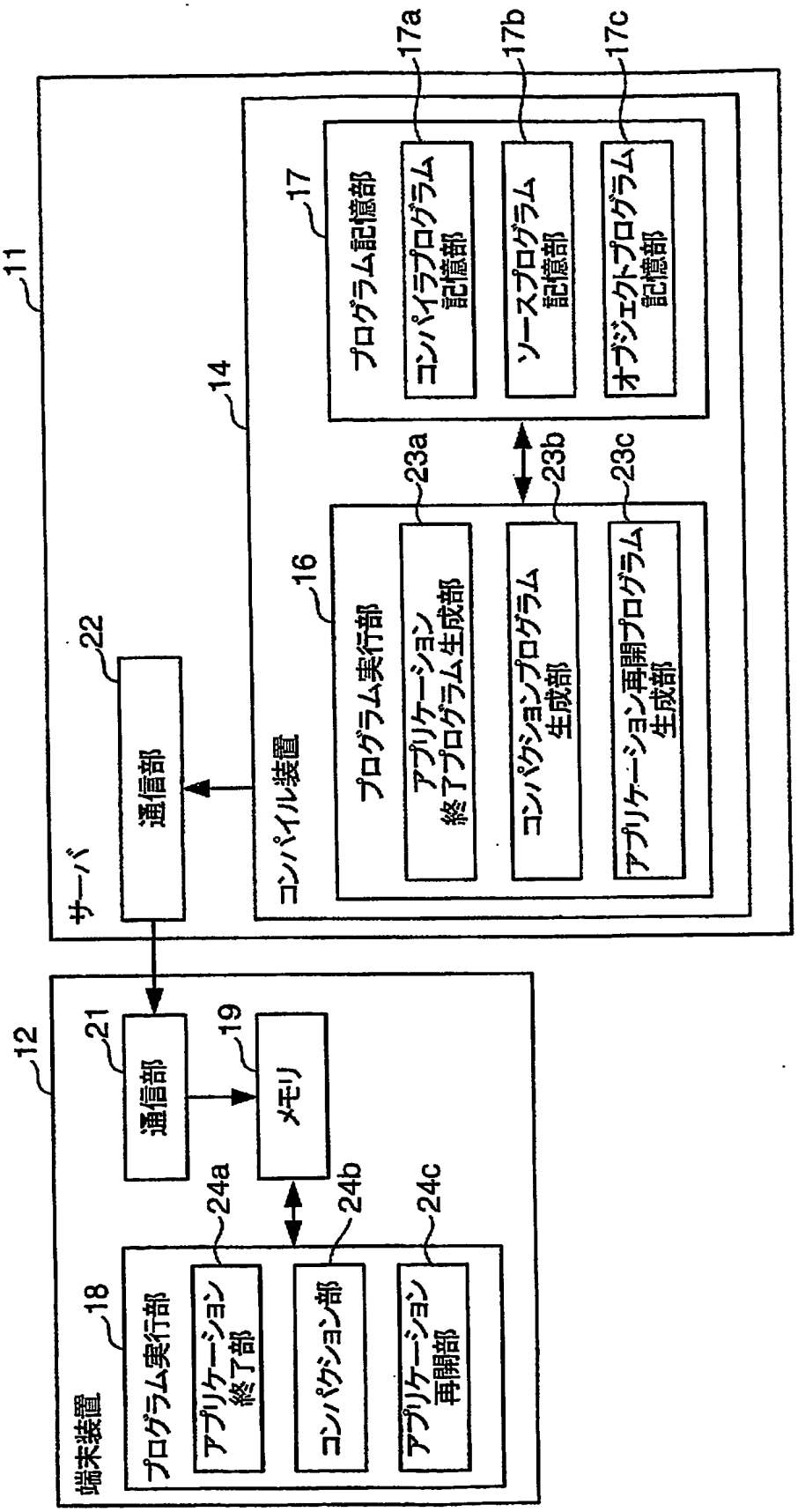
【図１８】


11
サーバ
22
通信部
14
コンパイル装置
12
端末装置
21
通信部
21a
識別番号付与部
20
データ読込部
20a
DPテーブル作成部
19
メモリ
19a
アプリケーションプログラムモジュール記憶部
19b
ライブラリプログラムモジュール記憶部
19c
スタック
19d
DPテーブル記憶部
18
プログラム実行部
18a
アドレス保存部
18g
識別番号取得部
18h
先頭アドレス取得部
18b
アドレス設定部
18c
移行部
18d
アドレス再設定部
18e
アクセス部

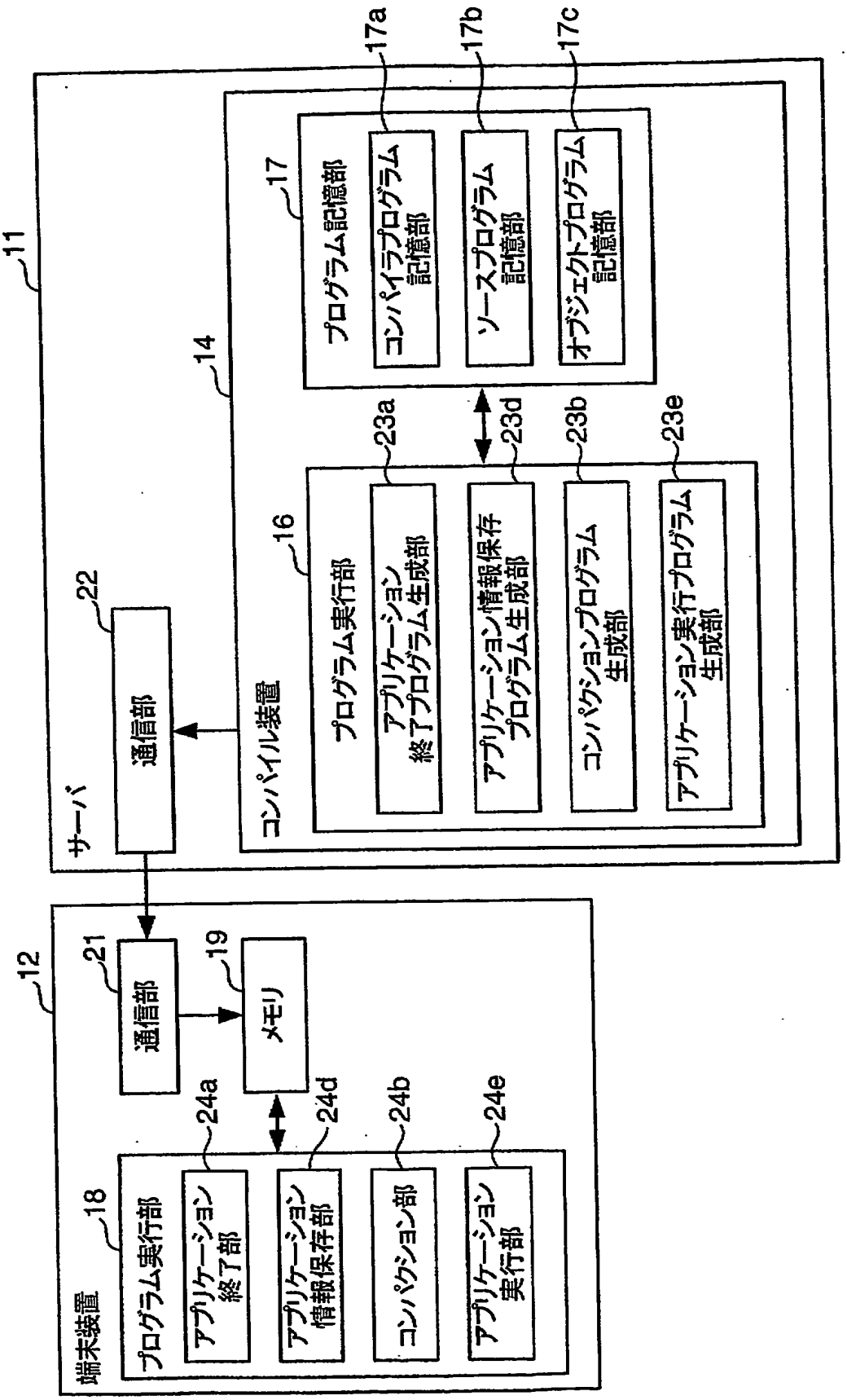
【図１９】


スタート
アプリケーションプログラムモジュール及び
ライブラリプログラムモジュールをダウンロード
S31
各プログラムモジュールに識別番号を付与
S32
各プログラムモジュールのコード領域を読み込む
S33
DPテーブルを作成
S34
各プログラムモジュールにデータ領域を割り当てる
S35
DPテーブルにデータ領域のアドレスを設定
S36
S37
アプリケーションプログラムモジュールから
ライブラリプログラムモジュールが
呼び出された?
NO
YES
アプリケーションプログラムモジュール
の先頭アドレスを退避
S38
ライブラリプログラムモジュールの識別番号を取得
S39
識別番号をインデクスとしてDPテーブルから
自モジュールのDPを取得
S40
取得したDPに切り替える
S41
アプリケーションプログラムモジュール
のデータ領域の先頭アドレスを復元する
S42
エンド

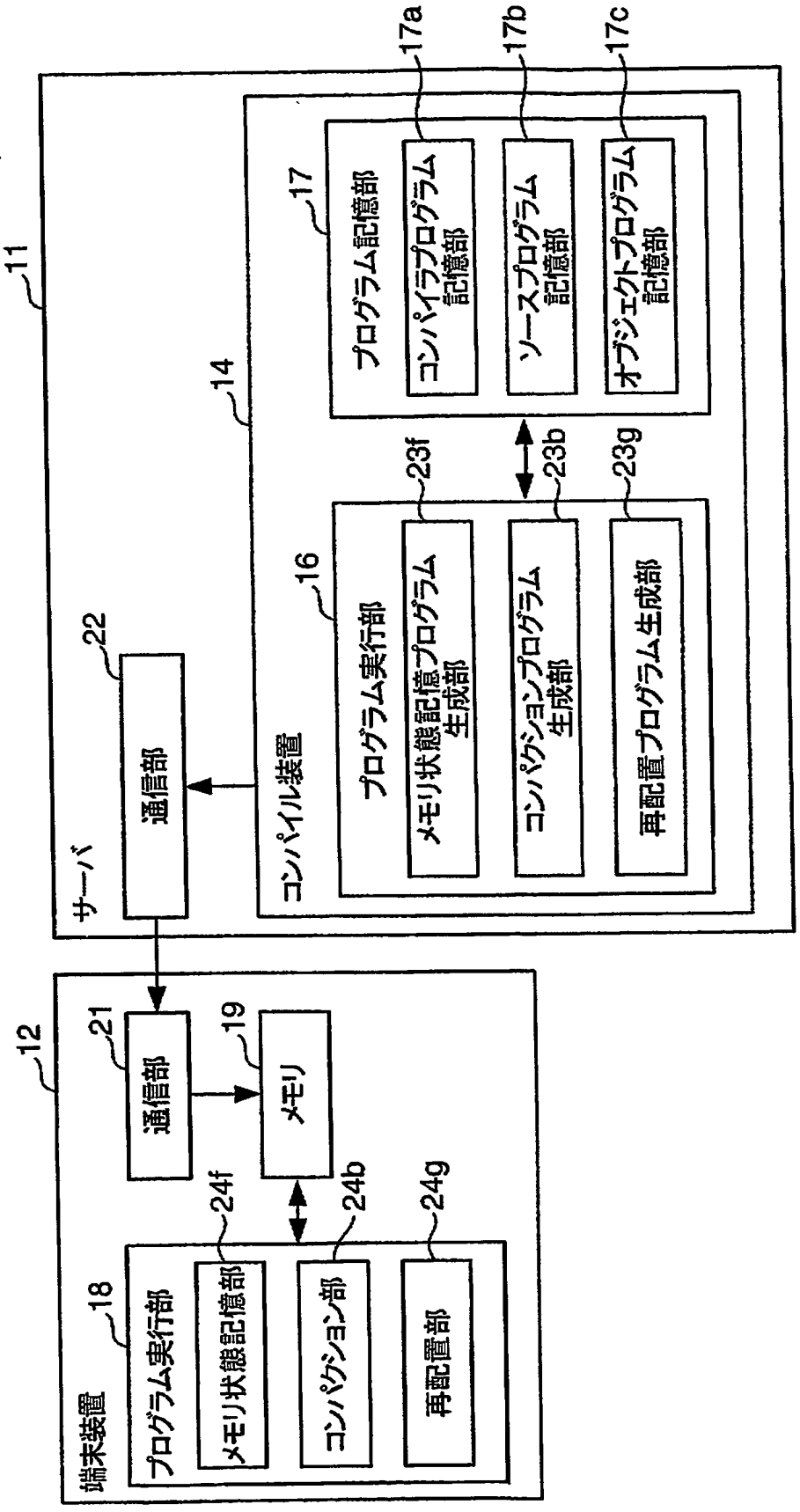
【図２０】


11
サーバ
22
通信部
14
コンパイル装置
16
プログラム実行部
23a
アプリケーション終了プログラム生成部
23b
コンパクションプログラム生成部
23c
アプリケーション再開プログラム生成部
17
プログラム記憶部
17a
コンパイラプログラム記憶部
17b
ソースプログラム記憶部
17c
オブジェクトプログラム記憶部
12
端末装置
21
通信部
19
メモリ
18
プログラム実行部
24a
アプリケーション終了部
24b
コンパクション部
24c
アプリケーション再開部

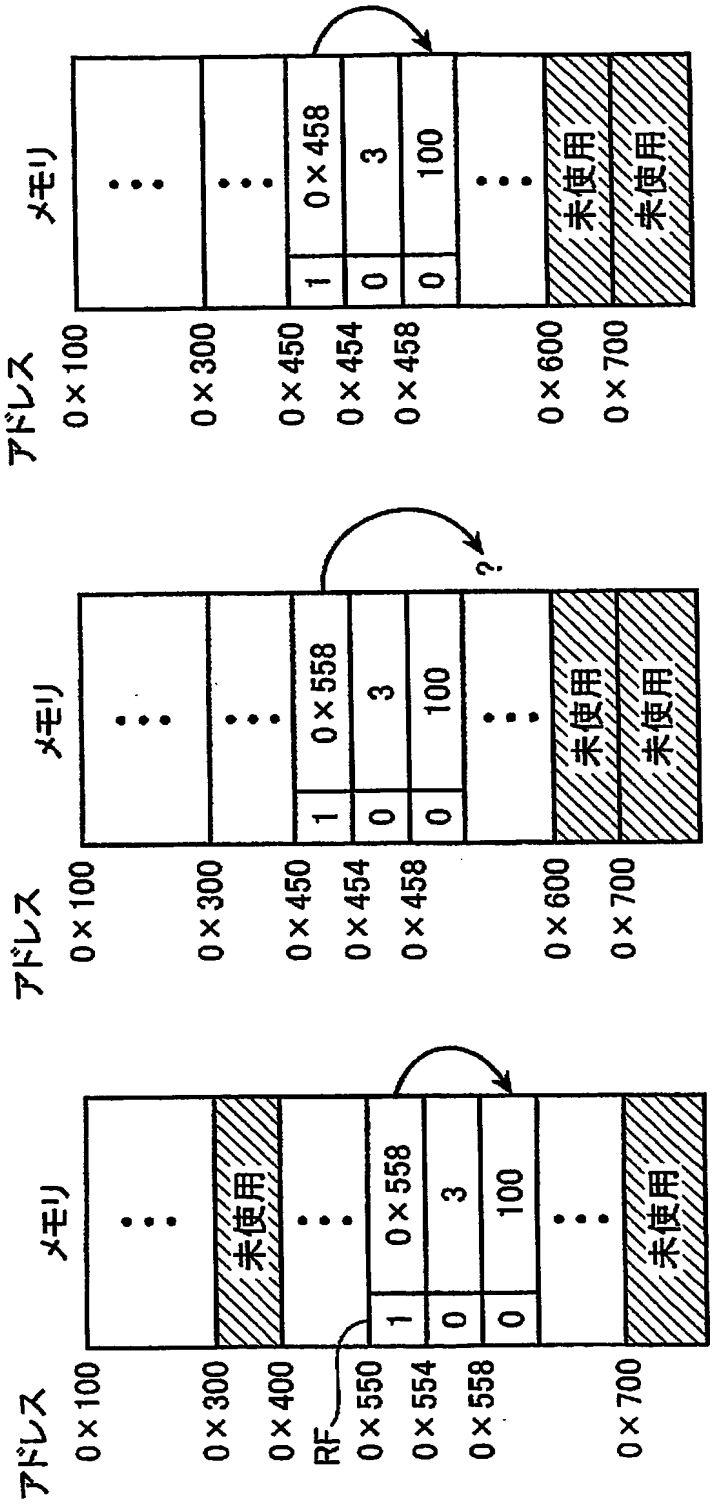
【図２１】


11
サーバ
通信部
22
14
コンパイル装置
16
プログラム実行部
23a
アプリケーション終了プログラム生成部
23d
アプリケーション情報保存プログラム生成部
23b
コンパクションプログラム生成部
23e
アプリケーション実行プログラム生成部
17
プログラム記憶部
17a
コンパイラプログラム記憶部
17b
ソースプログラム記憶部
17c
オブジェクトプログラム記憶部
12
端末装置
通信部
21
19
メモリ
18
プログラム実行部
24a
アプリケーション終了部
24d
アプリケーション情報保存部
24b
コンパクション部
24e
アプリケーション実行部

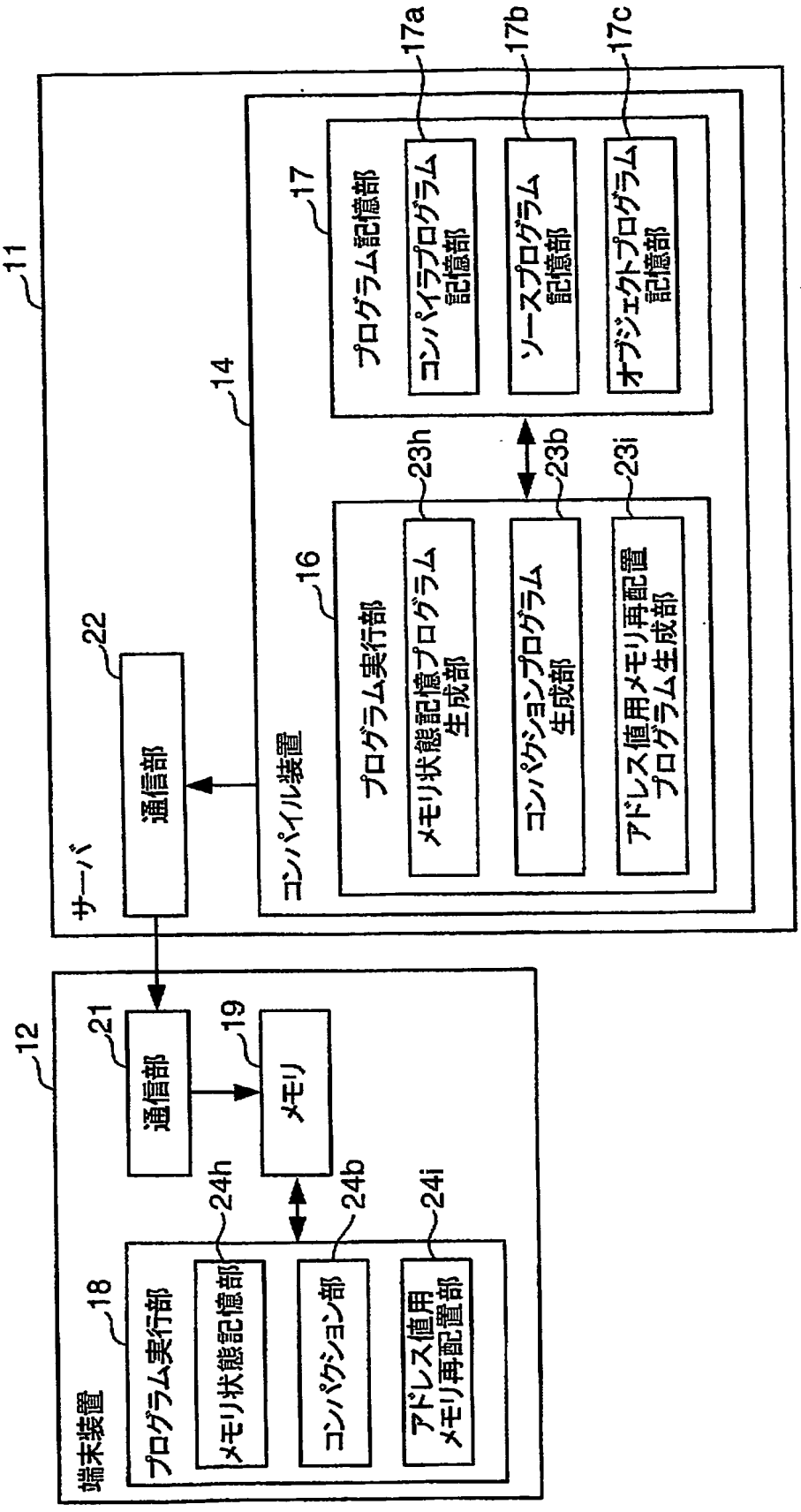
【図２２】


11
サーバ
22
通信部
14
コンパイル装置
16
プログラム実行部
23f
メモリ状態記憶プログラム生成部
23b
コンパクションプログラム生成部
23g
再配置プログラム生成部
17
プログラム記憶部
17a
コンパイラプログラム記憶部
17b
ソースプログラム記憶部
17c
オブジェクトプログラム記憶部
12
端末装置
21
通信部
19
メモリ
18
プログラム実行部
24f
メモリ状態記憶部
24b
コンパクション部
24g
再配置部

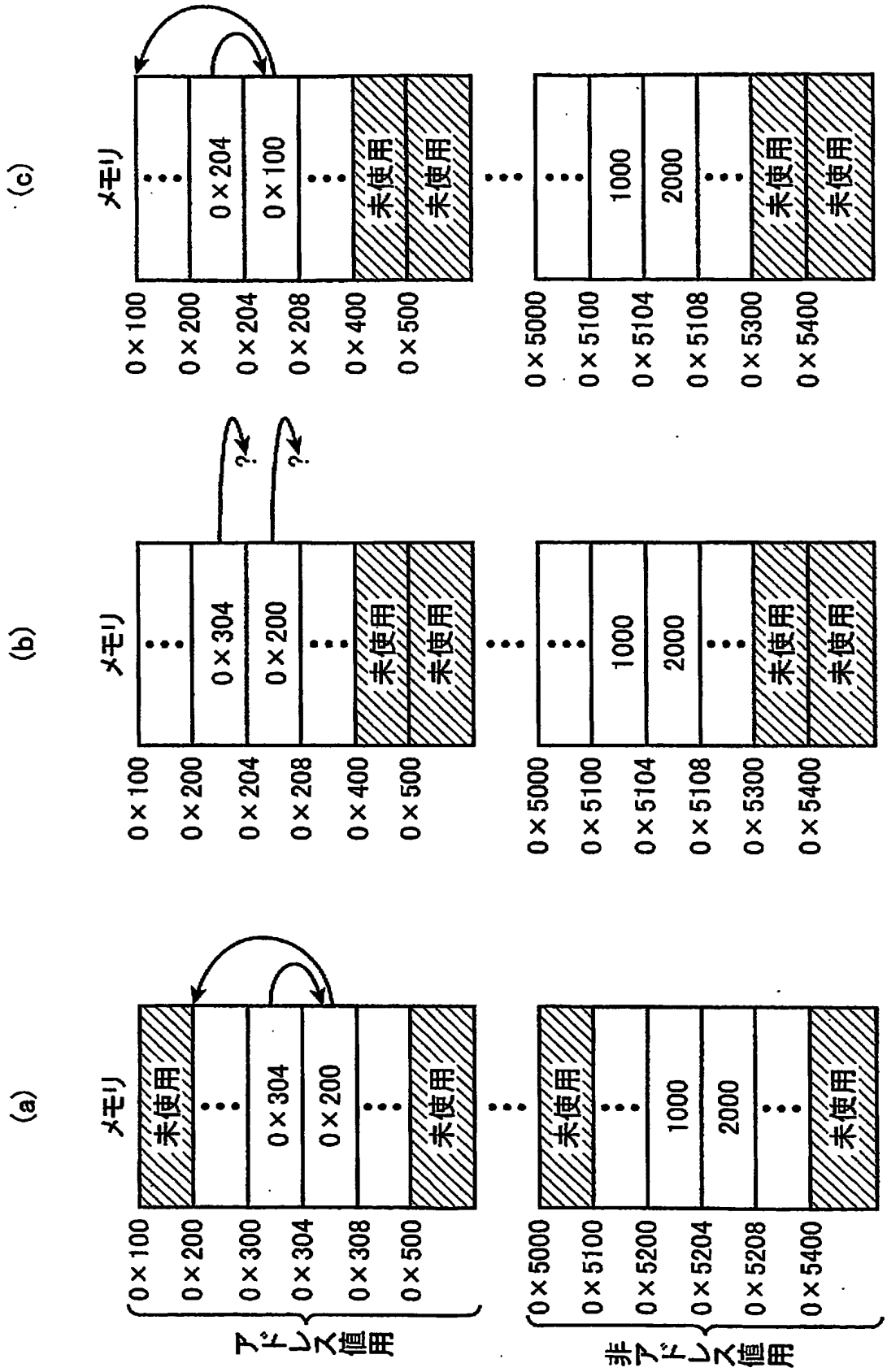
【図２３】


(c)
メモリ
0×458
3
100
1
0
0
未使用
未使用
アドレス
0×100
0×300
0×450
0×454
0×458
0×600
0×700
(b)
メモリ
0×558
3
100
1
0
0
未使用
未使用
?
アドレス
0×100
0×300
0×450
0×454
0×458
0×600
0×700
(a)
メモリ
未使用
0×558
3
100
1
0
0
未使用
アドレス
0×100
0×300
0×400
RF
0×550
0×554
0×558
0×700

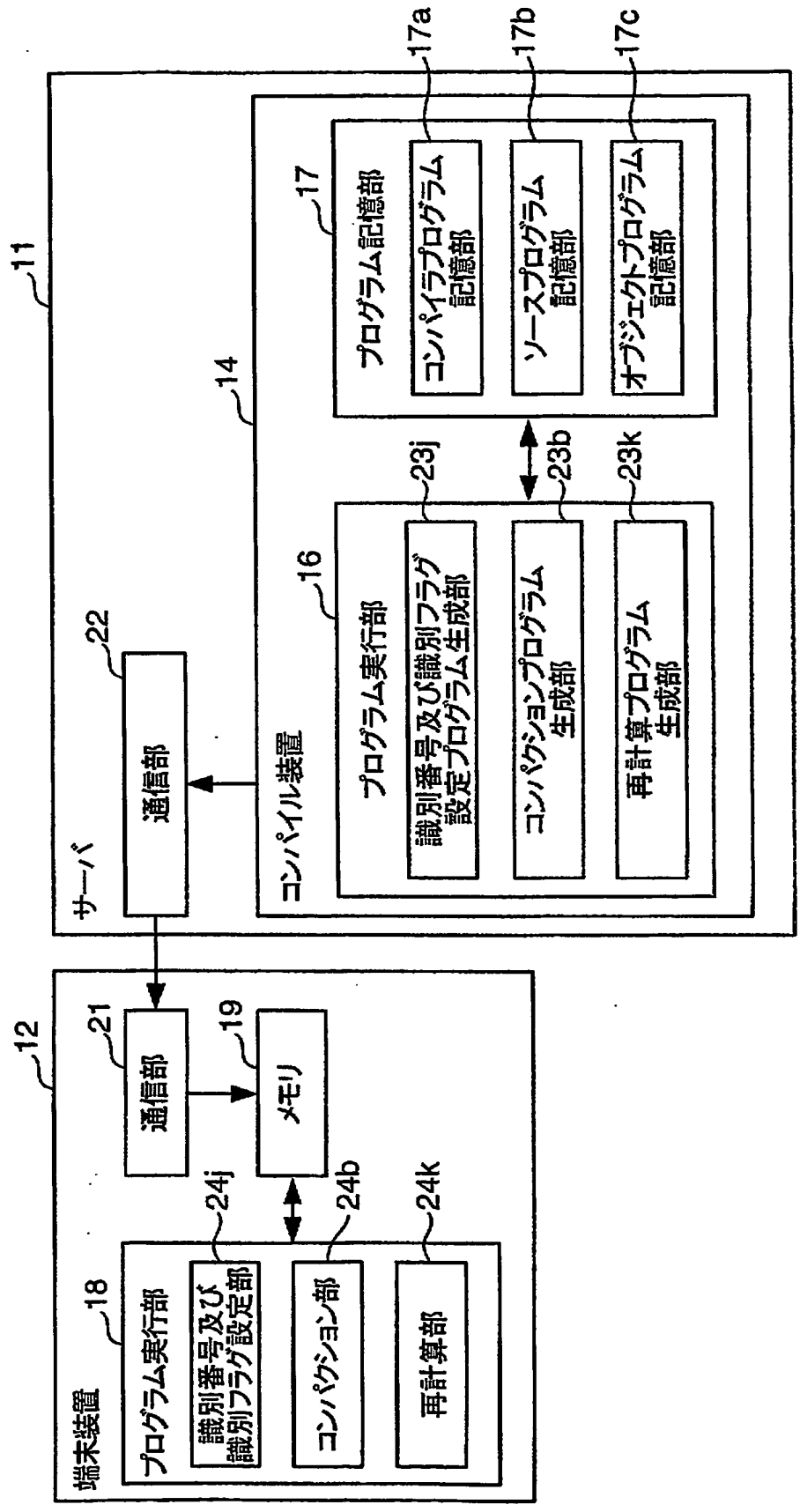
【図２４】


11
サーバ
22
通信部
14
コンパイル装置
16
プログラム実行部
23h
メモリ状態記憶プログラム生成部
23b
コンパクションプログラム生成部
23i
アドレス値用メモリ再配置プログラム生成部
17
プログラム記憶部
17a
コンパイラプログラム記憶部
17b
ソースプログラム記憶部
17c
オブジェクトプログラム記憶部
12
端末装置
21
通信部
19
メモリ
18
プログラム実行部
24h
メモリ状態記憶部
24b
コンパクション部
24i
アドレス値用メモリ再配置部

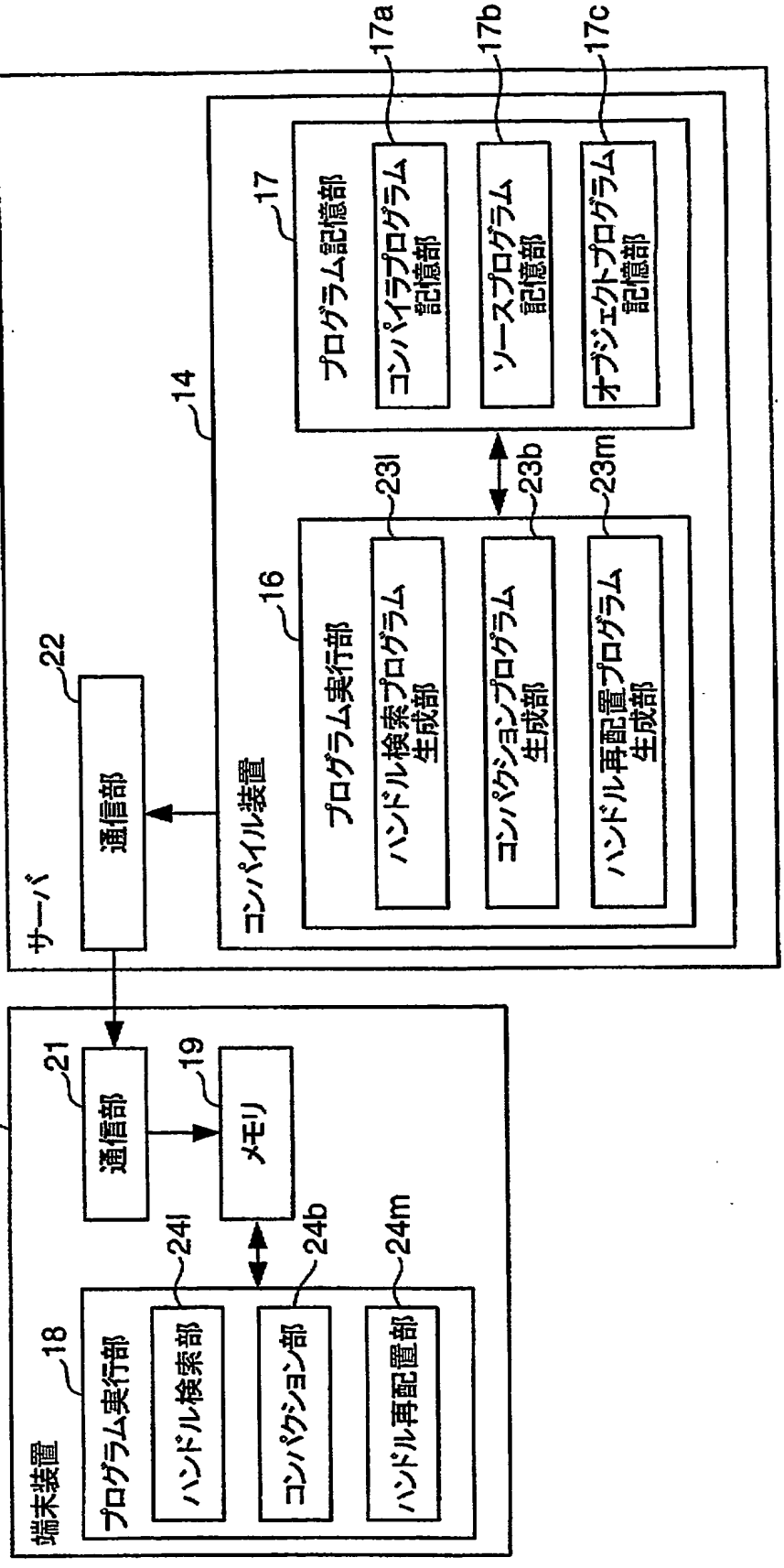
【図２５】


(c)
メモリ
未使用
0×204
0×100
1000
2000
0×100
0×200
0×204
0×208
0×400
0×500
0×5000
0×5100
0×5104
0×5108
0×5300
0×5400
(b)
メモリ
0×304
0×200
(a)
メモリ
0×304
0×200
0×300
0×304
0×308
0×5200
0×5204
0×5208
アドレス値用
非アドレス値用

【図２６】


11
サーバ
通信部
22
14
コンパイル装置
16
プログラム実行部
識別番号及び識別フラグ設定プログラム生成部
23j
コンパクションプログラム生成部
23b
再計算プログラム生成部
23k
17
プログラム記憶部
コンパイラプログラム記憶部
17a
ソースプログラム記憶部
17b
オブジェクトプログラム記憶部
17c
12
端末装置
通信部
21
メモリ
19
18
プログラム実行部
識別番号及び識別フラグ設定部
24j
コンパクション部
24b
再計算部
24k

【図２７】


11
サーバ
22
通信部
14
コンパイル装置
16
プログラム実行部
231
ハンドル検索プログラム生成部
23b
コンパクションプログラム生成部
23m
ハンドル再配置プログラム生成部
17
プログラム記憶部
17a
コンパイラプログラム記憶部
17b
ソースプログラム記憶部
17c
オブジェクトプログラム記憶部
12
端末装置
21
通信部
19
メモリ
18
プログラム実行部
241
ハンドル検索部
24b
コンパクション部
24m
ハンドル再配置部

【書類名】要約書
【要約】
【課題】　自動的に再入可能な目的プログラムを生成する。
【解決手段】　アドレス保存プログラム生成部１６ａは呼び出し元プログラムモジュールのデータ領域アドレスを保存するためのアドレス保存プログラムを生成し、アドレス設定プログラム生成部１６ｂは他プログラムモジュールのデータ領域アドレスを設定するためのアドレス設定プログラムを生成し、移行プログラム生成部１６ｃは呼び出し元プログラムモジュールから他プログラムモジュールへ移行するための移行プログラムを生成し、アドレス再設定プログラム生成部１６ｄは保存されたデータ領域アドレスを読み出して再設定するためのアドレス再設定プログラムを生成し、アクセスプログラム生成部１６ｅは設定されたデータ領域アドレスからの相対アドレスで他プログラムモジュールのデータ領域にアクセスするためのアクセスプログラムを生成する。
【選択図】　図２

特願２００３－４０２２０７

出　願　人　履　歴　情　報

識別番号　　[０００００５８２１]

１．変更年月日　　１９９０年　８月２８日
[変更理由]　　新規登録
住　所　　大阪府門真市大字門真１００６番地
氏　名　　松下電器産業株式会社